UNIVERSIDAD CARLOS III DE MADRID

ETIQUETADO DE ELEMENTOS EN ENTORNOS URBANOS MEDIANTE VISIÓN ESTÉREO Y REDES NEURONALES

Autor | Ramón Polo Herráez
Tutor | Basam Musleh Lancis

A todos aquellos que están junto a mí y lo han hecho posible.

Contenido

0. Listas

0.1.Lista de figuras

0.2.Lista de Tablas

1. Resumen

En la actualidad con el desarrollo de nuevas tecnologías en los procesadores y las cámaras digitales hace que el análisis de imágenes se haya vuelto más eficiente. Esto hace que se puedan procesar más información en menos tiempo.

Las grandes compañías de automóviles y otras compañías del ámbito tecnológico están desarrollando tecnología de detección de obstáculos. Esta tecnología dará una ayuda a la conducción muy importante ya que nos avisara de los posibles peatones y los obstáculos que pueden aparecer en la vía.

En este proyecto se desarrollara una red neuronal para la detección de objetos en las imágenes para ellos se partirá del mapa de disparidad, para después obtener el u_disparity de la imagen, este se etiquetará. Estos datos serán procesados y finalmente se incluirán en una red neuronal que utilizaremos para etiquetar las nuevas imágenes que la cámara estéreo detectará.

El objetivo es la obtención de una red neuronal capaz de diferenciar entre 5 categorías: peatones, vehículos, obstáculos, calzada y cielo. Esto ayudará a la hora de la conducción en los futuros coches, después se evaluará el nivel de acierto obtenido.

1.1.Abstract

Today with the development of new technologies in the digital cameras and processors makes the analysis of pictures more efficient. This means that it can process more information in less time.

The major car companies and other companies of the technology are developing obstacles detection technology. This technology will help us to drive, because it give us the information if there are pedestrians or obstacles at the road.

In this project we develop a neural network for detecting objects in images, for it I start from the disparity map, and then get the u_disparity of the image, this will label it. These data will be processed and eventually be included in a neural network that we use to label the new stereo camera images detected.

The goal is to obtain a neural network capable of differentiating between 5 categories: pedestrians, vehicles, obstacles, road and sky. This will help when driving in the future cars, then assess the level of success obtained.

2. Introducción

2.1.Motivación

En la actualidad una de las mayores causas de accidentes en la conducción se debe a colisiones con objetos que el conductor no ha visto, tanto por distracción o por falta de visibilidad. Estas distracciones según la DGT causan 4 de cada 10 accidentes de tráfico.[2]

Las estadísticas de mortandad de los últimos años de la DGT se pueden ver en la Fig.1.

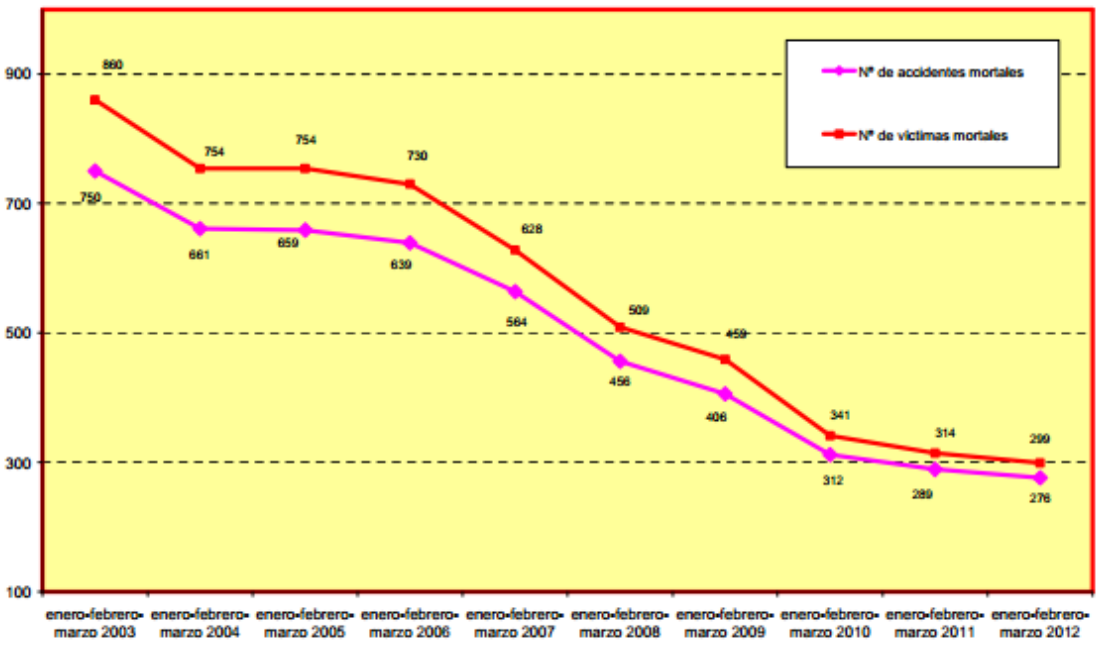


Figura 1 Relación temporal número de accidentes-número de víctimas mortales

Se puede observar como el número de víctimas y accidentes se va reduciendo pero si observamos con detalle los datos que se refieren al lugar de los accidentes, Fig.2.

Tipo de vía	Nº de accidentes con víctimas mortales			
	2011	2012	Distribución porcentual 2011	Distribución porcentual 2012
Autopista	7	18	2,4%	6,5%
Autovía	52	43	18,0%	15,6%
Vía para automóviles	4	0	1,4%	0,0%
Carretera convencional	204	207	70,6%	75,0%
Otras	22	8	7,6%	2,9%
TOTAL	**289**	**276**	**100,0%**	**100,0%**

Figura 2 Porcentajes de accidentes según tipo de vía.

La gran mayoría de accidentes se producen en carreteras convencionales, Fig.3, que son aquellas en las que se circula en ambos sentidos, y puede haber peatones caminando por los arcenes ya que estas carreteras se suelen ubicar entre pequeñas localidades.

Figura 3 Carretera convencional

Los accidentes en este tipo de carreteras pueden ser tanto colisiones frontales con coches, colisiones con obstáculos en los laterales de la carretera o atropellos de peatones, en el siguiente gráfico, Fig.4, podemos observar cómo se distribuyen las víctimas de accidentes.

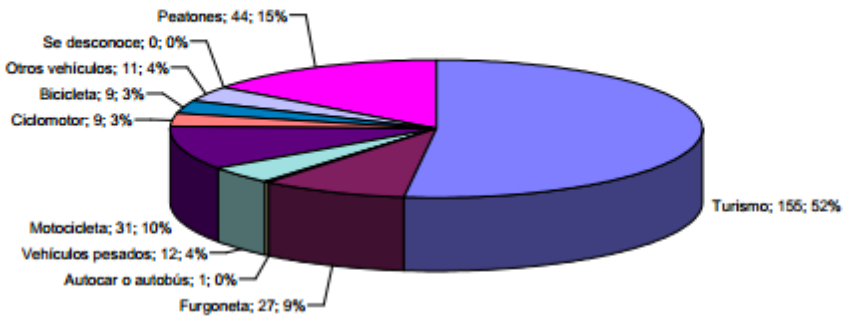


Figura 4 Porcentajes de victimas según medio de transporte

Observamos que la mayoría de accidentes son con turismos pero la segunda mayor causa de víctimas son aquellas situaciones donde se ven involucrados peatones. El fin de este proyecto es la detección y clasificación de objetos en los entornos de tráfico, para que el conductor tenga una mejor información de los elementos del entorno de conducción, con lo que se podría reducir el número de accidentes. Pudiendo ser una herramienta muy útil en futuras implementaciones de conducción autónoma.

2.2.Sistemas de visión asociados a seguridad y transporte.

En la actualidad se pueden observar una mayor informatización del mundo, los coches llevan ordenadores de a bordo que son más potentes que los ordenadores de sobre mesa de hace 15 años, toda esta evolución hace que se puedan tener sistemas inteligentes de transportes tanto a nivel de automóvil, como transporte público (autobuses, trenes, …). Estos sistemas se denominan ADAS (Advanced Driver Assistance System), usan interfaces humano-maquina para conseguir un aumento de la seguridad.

Está evolución ha dado lugar a un mundo totalmente informatizado que está empezando a introducir los sistemas de visión en su día a día, algunos ejemplos de la detección de objetos son:

2.2.1. Google Car

Google [4] ha equipado una flota de al menos 10 coches para las pruebas en carretera de su coche de conducción autónoma, estos coches son 6 Toyota Prius, 3 Lexus RX450h, Fig.5, y un Audi TT, estos coches están equipados con un sistema de localización GPS y un radar laser montado en el techo el cual proporciona información 3D del entorno.

Estos coches ya han sido probados en condiciones de tráfico normal, obteniendo buenos resultados sin ningún accidente por el momento, el equipo anunció que habían conseguido 500 000 km de conducción autónoma sin accidentes.

Figura 5 Lexus RX450h equipado con el sistema de conducción autónoma de Google

2.2.2. Detección automática de conductas sospechosas en aeropuertos.

Otro ámbito relacionado con la detección por visión por ordenador es la seguridad civil. Creando programas y aplicaciones que detectan ciertos factores en las personas

se puede incrementar la seguridad de los aeropuertos. Este tipo de programas es una ayuda para el personal de seguridad de los aeropuertos, estaciones de tren y diversos lugares públicos como ayuntamientos, calles muy concurridas etc.

Estos sistemas se basan en distintas metodologías y sistemas siendo uno de los más usados la detección facial de personas mediante video vigilancia, Fig.6, estos programas se basan en detectar primero donde se encuentra la cara y después analizar una serie de medidas biométricas, debido a la alta complejidad del análisis de rasgos humanos el resultado no es 100% seguro, pero es lo suficientemente preciso para enviar a un agente a verificar si se trata de la persona que están buscando.

Figura 6 Sistema de detección facial

Otra utilidad es la detección de objetos abandonados, Fig.7, principalmente estos sistemas se basan en el restado del fondo de la imagen, etiquetando los objetos y observando sus movimientos, una vez que un objeto se divide y uno de los nuevos objetos queda abandonado este se marcara como peligro. Esta implementación de seguridad está muy extendida ya que en los últimos años se ha puesto mucho énfasis en la detección de posibles atentados terroristas.[5]

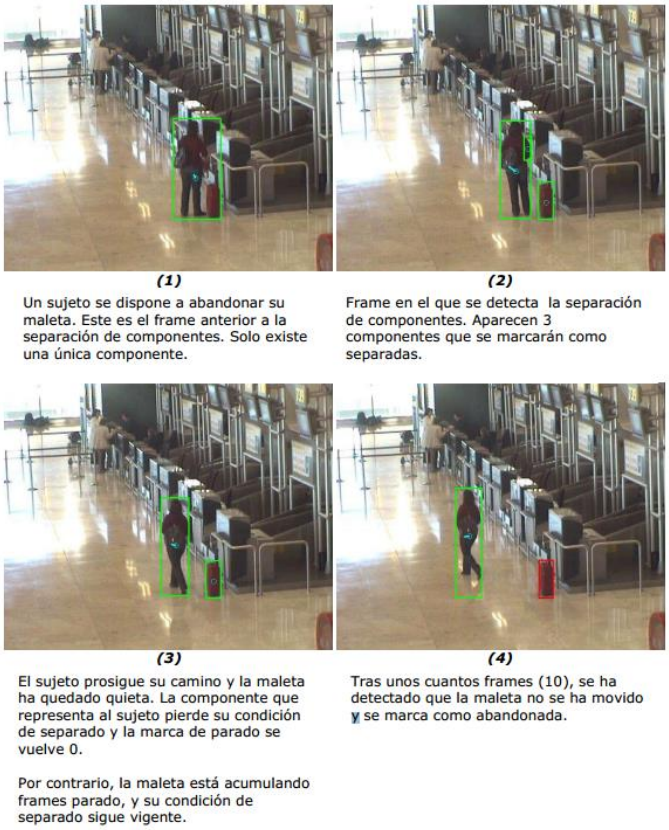


Figura 7 Sistema de detección de objetos abandonados

2.3.Nuevas líneas de investigación en sistemas de visión estéreo.

En la actualidad se están llevando acabado distintas investigaciones en el campo de la detección y clasificación de objetos mediante sistemas de percepción. Esta área avanza cada vez más deprisa gracias a los nuevos procesadores y métodos de programación, esto hace que se puedan manejar más datos a mayor velocidad. En los sistemas de detección de objetos mediante análisis de imágenes hay un gran volumen de datos.

A continuación se discutirá y explicará las últimas tendencias en el campo de la detección de obstáculos:

2.3.1. Urban 3D Semantic Modelling Using Stereo Vision.[13]

Este artículo de la universidad de Oxford describe un algoritmo robusto basado en el análisis de imágenes estéreo, estas imágenes se usaran para crear un mapa semántico en 3 dimensiones del entorno urbano. Este algoritmo se usará para el

guiado de los robots, ya que actualmente muchos de los robots están guiados por sistemas basados en laser los cuales dan muy poca información acerca del ambiente, y comparados con las cámaras, los sensores laser son muy caros y usan mucha energía, reduciendo la autonomía del robot.

Este nuevo modelo usando cámaras en lugar de láser, es muy ventajoso ya que se utilizará la información pixel por pixel, en lugar de la información dispersa que nos ofrece un láser, esto también hace que la imagen tenga más detalles y nos permite tener una mejor definición del entorno.

En el ámbito de la clasificación de los objetos se ha comprobado eficientemente que la clasificación de imágenes de carreteras tiene un alto índice de acierto, en este caso el etiquetado de los objetos también se ha utilizado para generar un mapa cenital del recorrido efectuado por el coche.

Este artículo escrito por Sunando Sengupta, Eric Greveson, Ali Shahrokni y Philip H. S. Torr describe como usando una serie de algoritmos matemáticos se puede crear una descripción del entorno, para esto usarán como datos de entrada una secuencia de imágenes calibradas. Estas parejas de imágenes rectificadas hacen que las líneas de escaneo de la imagen sean las líneas epipolares de la imagen estéreo. La posición estimada de la cámara se usara para generar una representación volumétrica del entorno. Conectando estos datos a la red se pueden crear grandes reconstrucciones de las calles. A la vez se usa el modelo CRF (Conditional Random Field) para etiquetar los objetos de la imagen.

2.3.1.1. Reconstrucción semántica 3D del mundo.

Aquí describen las fases de la reconstrucción.

2.3.1.1.1. Reconstrucción de la superficie.

Para reconstruir la imagen primero realizan un mapa de profundidad a partir de las parejas de imágenes. Para esto utilizan el TSDF (Truncated Signed Distance Function).[13]

Para la generación de este mapa la profundidad será calculada con

$$z_i = {B \cdot f}/{d_i} \tag{2.3.1}$$

Donde z es la profundidad, d la disparidad del pixel i, B es el baseline de la cámara y f la distancia focal. Para esto usan la implementación OpenCV del SGBM (Semi-Global Block Matching).

Con el TSDF se realizara la estimación del volumen, para esto se utilizará una función con signo en la cual los valores positivos corresponden a espacio libre, +μ, y los valores negativos corresponden con espacio detrás de la superficies, - μ. Asumen que los valores reales estarán entre los valores ±μ, cualquier valor por fuera de este rango es ignorado.

Como también se realiza una actualización del volumen para la reconstrucción del trazado, para cada nuevo mapa de profundidad la red se actualizará.

Para poder obtener una superficie unificada se creará una red triangulada usando el algoritmo Maching Tetrahedra [17], con este algoritmo se obtendrá la red triangulada de la superficie-iso de valor cero. El modelo reconstruido tiene muchos detalles que permitirán distinguir entre objetos, una vez obtenidos estos valores se realizará el etiquetado. Este tipo de etiquetado hace que el sistema sea más rápido que etiquetar pixel por pixel en 3D y aun produce un etiquetado lo suficientemente denso.

2.3.1.1.2. Generación del modelo semántico.

Para la creación de este modelo se usa una aproximación basada en un CRF (Conditional Ramdom Field) [18], este sistema considera una serie de variables

aleatorias que toma el valor de una etiqueta de un entramado alrededor de un pixel (v), este subconjunto de pixeles serán sus vecinos, se pueden usar tanto los vecinos con distancia 4 como con distancia 8.

Para obtener la segmentación y el etiquetado de estas usaremos el CRF y la ecuación de la energía de Gibbs:

$$E(x) = \sum_i \psi_i(x_i) + \sum_{i,j} \psi_{ij}(x_i, x_j) + \sum_{i,j} \psi_{ij}^d(x_i, x_j) + \sum_c \psi_c(x_c) \quad (2.3.2)$$

Donde ψ_i es el potencial unitario, que representa el coste de aplicar una etiqueta a un pixel, y ψ_{ij} es el potencial de la pareja que hace que el etiquetado sea más suave, ψ_{ij}^d es el potencial de disparidad y por último ψ_c el potencial de orden superior que define el potencial de solapar superpixels.

Para terminar el etiquetado se hace una fusión de etiquetado, se seleccionan una serie de puntos aleatorios. Después estos puntos se proyectaran sobre una serie de imágenes y se compararan de manera que la etiqueta que este más repetida será la designada para ese punto

.

2.3.1.2. Experimentos.

Finalmente realizaron una serie de experimentos con imágenes de 1241x376. Se utilizaron 45 imágenes para el entrenamiento y 25 para el test pixel por pixel, las etiquetas que ellos usaron son road, vehicle, pedestrian, pavement, tree, sky, signage, post/pole, wal/fence.

Para evaluar los resultados se consideró el error de traslación y el de rotación, cuanto más frames usan menos error obtienen. En cuanto a la velocidad, al usar el método completo les lleva unos 3,5 segundos por frame, mientras que mediante el método rápido obtienen unos 4 fps.

2.3.1.3. Conclusiones.

Finalmente concluyen que han obtenido un innovador sistema basado en visión estéreo, que produce un buen etiquetado de la imagen, así como un buen seguimiento de la ruta realizada.

2.3.2. Detección de peatones usando imágenes estéreo y señales 2D en un vehículo en movimiento.[14]

En este artículo de la universidad de Henan y escrito por Yang Yang, Jingyu Yang y Dongyan Guo se describe un sistema de detección de peatones con cámaras de bajo coste, esto lo realizan usando una mapa de superficie parallax y un clasificador SVM (Support Vector Machine). El objetivo es conseguir un sistema de detección de peatones eficiente y barato para el mercado del automóvil en China, y utilizan las cámaras estéreo, debido a que solo con la información dada por un sistema 2D no es suficiente para detectar las formas complejas de los peatones. Esta detección se realiza sacando información de los mapas de disparidad.

Para la detección de peatones utilizan ROI (Regions Of Interest), esto hace que el área donde se buscan a los peatones este delimitada. Esto hace que no se tenga que realizar una búsqueda intensiva en toda la imagen sino en solo una parte de la imagen. Se realiza dividiendo las imágenes en zonas más pequeñas usando los discontinuidades en los mapas de disparidad. Para esto se usa un sistema GAVRILA [19], que utiliza un sistema de multiresolución para generar las ROIs. Las aproximaciones de multiresolución se realizan buscando relaciones a nivel general que después se pueden refinar. También se usará este sistema para obtener el área de la carretera, y después se realizara una reconstrucción del mundo tridimensional. Para reducir el espacio de búsqueda solo los bordes de las imágenes derecha-izquierda son elegidos en la reconstrucción del modelo estéreo.

2.3.2.1. Generación de ROI basadas en SPM

Esta aplicación utiliza una versión diferente de los mapas de disparidad en este caso solo se utilizará la zona de la carretera, esta variante se denomina en inglés como Surface Parallax Map (SPM), Fig. 8. Considerando que los peatones y otros objetos no pertenecen a la carretera y están localizados entre esta y la cámara.

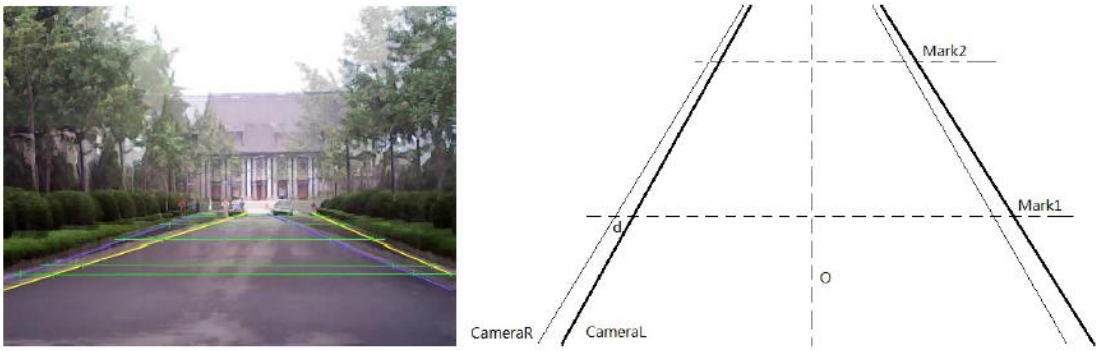


Figura 8 SPM

Normalmente los métodos que se utilizan para la detección están basados en las coordenadas de la imagen, pero con el método que proponen se podría detectar peatones en un rango de unos 50 metros con cámaras de bajo coste. Esto lo consiguen introduciendo GCPs (Ground Control Point), estos se usaran para ayudar a conseguir la paridad de los pixeles de la carretera y conseguir el SPM.

El SPM se consigue sabiendo que las imágenes obtenidas provienen de unas cámaras estéreo fijadas con cierto ángulo, estas imágenes se toman al mismo tiempo. Como se puede ver en el diagrama de la parte derecha de la Fig. 8, tenemos una linea O la cual paso por el punto medio de las dos imágenes. Los puntos GCPs nos dan la posición de la carretera, para esto se deberán fijar unas líneas cuando las cámaras se calibra, para evitar que las líneas del parallax sean verticales. Considerando que la carretera está enfrente del vehículo, esta se puede asimilar a un plano. Esto hace que las líneas parallax y la distancia a la cámara tengan un correlación linear. Como utilizan cámaras de 320x240 esto hace que los detalles no sean muy precisos y presentan una limitación en la distancia de detección, para poder tener unas líneas parallax de 50 metros , los dos ejes ópticos deberán cruzarse detrás de la cámara.

Para obtener esta líneas se utiliza la siguiente formula:

$$d_y = SPM(y) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y-\Delta y)\frac{1}{m}\sum_{j=1}^{m} d_j}{y_i - \Delta y} \qquad (2.3.3)$$

Donde *y* es la coordenada ordenada de la imagen, Δy es la ordenada en el origen, m el número de líneas de control, n es el número de GCPs e y_i es la coordenada ordenada del GCP i-esimo.

2.3.2.2. Corrección en tiempo real de los SPM.

En este apartado intentan corregir la suposición de que todas las carreteras están en el mismo plano cuando en realidad tiene curvas y cambios de nivel. Como las cámaras están fijas al coche las rotaciones en el eje Y y Z no afecta a los SPM pero la rotación sobre el eje X sí que tiene efectos sobre estos. Para esto deciden utilizar el detector de CANNY para localizar los bordes de la carretera. Después de calcular la oclusión y proyectar sobre el eje YOZ se obtiene el ángulo β que es el ángulo entre la horizontal y la carretera, Fig. 9.

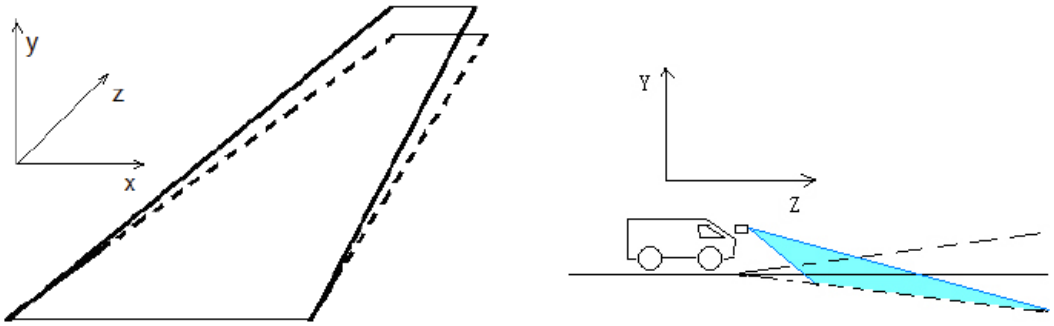


Figura 9 Imagen de la inclinación de la carretera.

A continuación podemos ver como haciendo la proyección YOZ del mapa de disparidad podemos obtener el ángulo, Fig. 10.

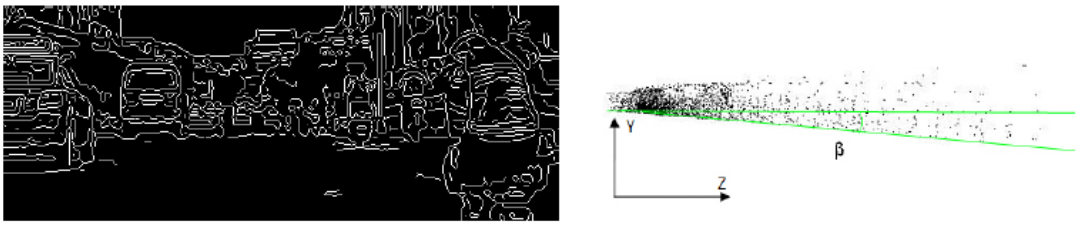


Figura 10 Nube de disparidad

Y por último se corrige la ecuación de parallax agregando el factor de corrección s:

$$d_y = SPM(sy) = \frac{1}{n}\sum_{i=1}^{n}\frac{(sy-\Delta y)\frac{1}{m}\sum_{j=1}^{m} d_j}{y_i - \Delta y} \qquad (2.3.4)$$

Donde s es el factor de corrección que se calcula de la siguiente manera:

$$s = \frac{\tan\alpha}{\tan\alpha + \tan\beta} \qquad \tan\alpha = \frac{H}{L} \qquad L = \frac{1}{n}\sum_{i=1}^{n}\frac{y-\Delta y}{y_i - \Delta y}L_i \qquad (2.3.5)$$

β es el ángulo con la horizontal, H es la altura donde a la que está colocada la cámara del suelo, L es la distancia de la superficie de la carretera.

2.3.2.3. Verificación de los peatones en las ROI.

La detección de los peatones se realiza mediante la detección de la forma y la textura. Se utiliza el clasificador Real AdaBoost combinado con Haar wavelets y Edge orientation histograms. Durante el entrenamiento el tamaño de la ventana que se utiliza es de 128x64 y cada celda es de 8x8, el histograma se calcula para cada celda y un bloque se considera como 2x2 celdas y el algoritmo L2-Hys es calculado.

2.3.2.4. Experimento

En el experimento se utilizan una serie de secuencias de video grabadas en el tráfico urbano, para ello utilizan una procesador Intel Core2 2.0GHz con 2 Gb RAM y

cámaras de bajo coste con un rango de 50 metros y un ángulo de visión de 50 grados. Aplicando todos los algoritmos finalmente obtienen los siguientes resultados:

Algorithm	Max hit rate	Max effective distance	Processing time
SPM-based	96%	50 m	3 fps
Stereo-match-based	60%	15 m	0.4 fps

2.3.2.5. Conclusión.

Finalmente concluyen que el algoritmo es suficientemente robusto y preciso, ya que realizan numerosas pruebas sobre distintas secuencias de video.

3. Fundamentos teóricos

Para este proyecto se realizará el etiquetado de calzada, cielo, peatones, vehículos y obstáculos usando imágenes estéreo y una red neuronal creada en Matlab. Que mediante el etiquetado de unas imágenes de muestra será capaz después de un entrenamiento de hacer la diferenciación entre los elementos de la vía.

Se partirá desde un mapa de disparidad de las imágenes estéreo para obtener el u_disparity. A partir de la información el u_disparity, se etiquetaran las imágenes. Y con los datos obtenidos de este etiquetado, se entrenará la red que nos dará los resultados.

En este apartado se explicaran brevemente los fundamentos teóricos de los principios ópticos, mapas de disparidad y redes neuronales necesarios para la implementación del proyecto.

3.1.Principios ópticos

Una breve introducción sobre la captación de la imágenes estéreo y como se procesan.

3.1.1. Modelo de lente fina

En este modelo se supone que la lente tiene un grosor despreciable, esta suposición hace que los cálculos sean muchos más sencillos y rápidos. Aunque al ser una aproximación se cometan errores.

La luz incide sobre la lenta de manera perpendicular a esta y debido a la refracción de la luz, la lente hace que todos estos rayos se desvíen de su trayectoria, Fig. 11. Todos se concentran en un punto, llamado el foco. Los únicos rayos que no verán alterada su trayectoria serán todos aquellos que pasen por el centro óptico de la lente, estos rayos de luz no se verán alterados por la lente.

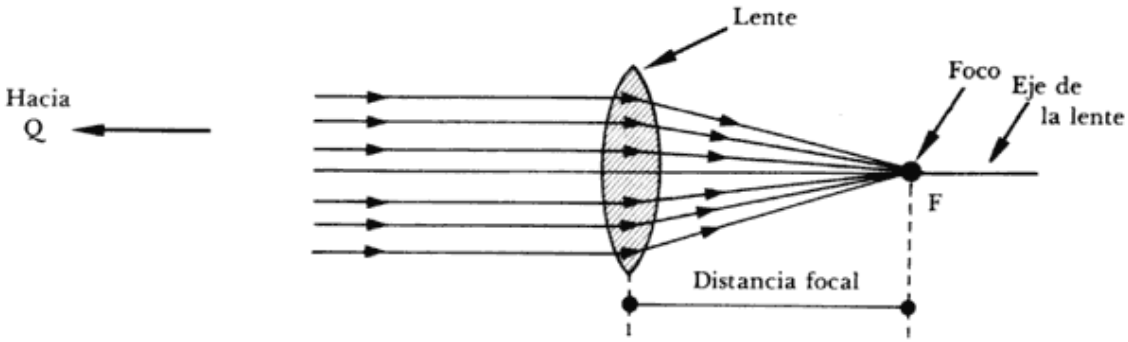


Figura 11 Modelo lente fina [6]

3.1.2. Modelo Pin-Hole

Este modelo se basa en el cámara estenopeica [7], estas cámaras fueron las primeras en surgir y se basan en una caja oscura con un material fotosensible en su interior. A esta caja se le realiza un pequeño orificio (0,5 mm) para obtener una buena definición de imagen, Fig. 12. Para estas cámaras se necesitaba un tiempo de exposición considerable. Pero lo que nos da este modelo es que aporta sencillez, lo que simplifica muchos los cálculos a realizar.

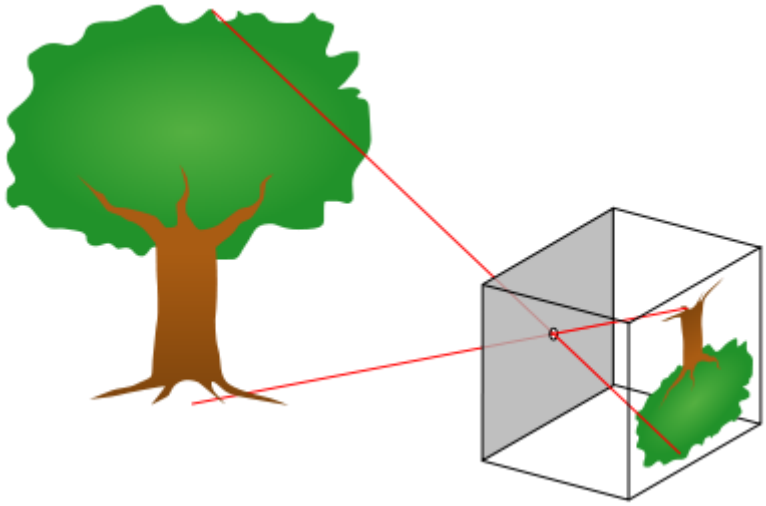


Figura 12 Modelo Pin-hole

Mediante trigonometría se pueden obtener las relaciones entre las coordenadas del mundo real (P(W,U,V)), con la coordenadas bidimensionales de la fotografía (p(u,v)).

$$u = \frac{f}{W}U$$

$$v = \frac{f}{W}V \tag{3.1.1}$$

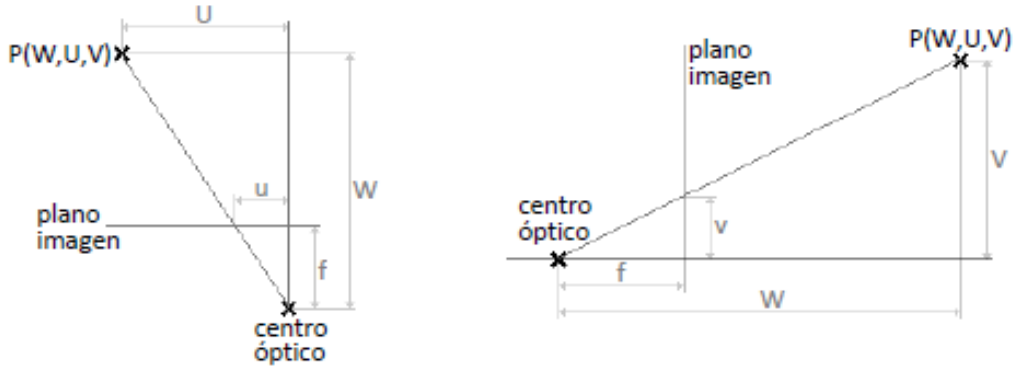


Figura 13 Representación trigonométrica de los puntos en el espacio. (A) y (B)

En la Fig. 13 (A) se puede observar la vista cenital del modelo de la cámara y en la Fig. 13 (B) la vista lateral.

3.1.3. Visión estéreo

Para la visión estéreo se utilizan dos cámaras, para el modelo ideal se considera que los dos ejes ópticos [9] de cada cámara son paralelos y tienen la misma distancia focal. Con la visión estéreo lo que se obtiene es que para un mismo punto se obtienen 2 proyecciones distintas de cada camara, como ejemplo si tenemos un punto perteneciente al mundo real P(W,U,V), al tener dos imágenes obtenidas desde puntos distintos obtendremos unos puntos $p_1(u_1,v_2)$ y $p_2(u_2,v_2)$. Normalmente estos puntos no coinciden en las coordenadas, Fig. 14.

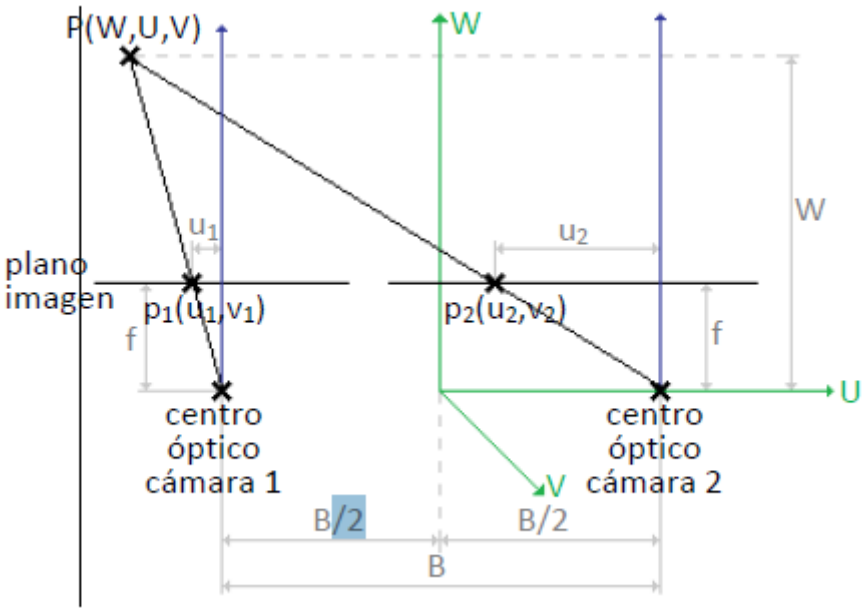


Figura 14 Sistema de visión estéreo

Si aplicamos las ecuaciones del modelo pin-hole al sistema estéreo, se obtienen las siguientes ecuaciones:

$$\frac{u_1}{f} = \frac{U - B/2}{W} \tag{3.1.2}$$

$$\frac{u_2}{f} = \frac{U + B/2}{W} \tag{3.1.3}$$

Al despejar las dos ecuaciones anteriores se puede obtener la componente W (profundidad) que con una cámara sencilla sería muy complicado de obtener.

$$\frac{u_1 \cdot W}{f} + \frac{B}{2} = U \tag{3.1.4}$$

$$\frac{u_2 \cdot W}{f} - \frac{B}{2} = U \tag{3.1.5}$$

$$\frac{u_1 \cdot W}{f} + \frac{B}{2} = \frac{u_2 \cdot W}{f} - \frac{B}{2} \tag{3.1.6}$$

$$\frac{u_2 \cdot W}{f} - \frac{u_1 \cdot W}{f} = B \tag{3.1.7}$$

$$\frac{W}{f}(u_2 - u_1) = B \tag{3.1.8}$$

$$W = \frac{B \cdot f}{(u_2 - u_1)} \tag{3.1.9}$$

Esta última expresión nos da la profundidad con lo que ya podemos obtener todas las coordenadas del mundo real.

3.1.4. Geometría epipolar

Este tipo de geometría permite relacionar las imágenes tridimensionales con sus proyecciones en las imágenes. Tenemos dos camaras (O_L,O_R), Fig. 15, que están separados una distancia B, que se denomina línea de base. Sus ejes ópticos son paralelos y tenemos que un punto P en el espacio tridimensional.

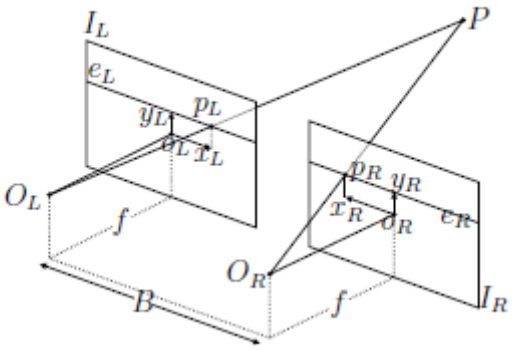


Figura 15 Geometría epipolar

Como se puede observar en la imagen la proyección del mismo punto en las imágenes es distinta. Estas diferencias nos ofrecen mucha información acerca de la disposición tridimensional de los objetos. En la Fig. 16 se observa con más claridad la diferencia entre imágenes.

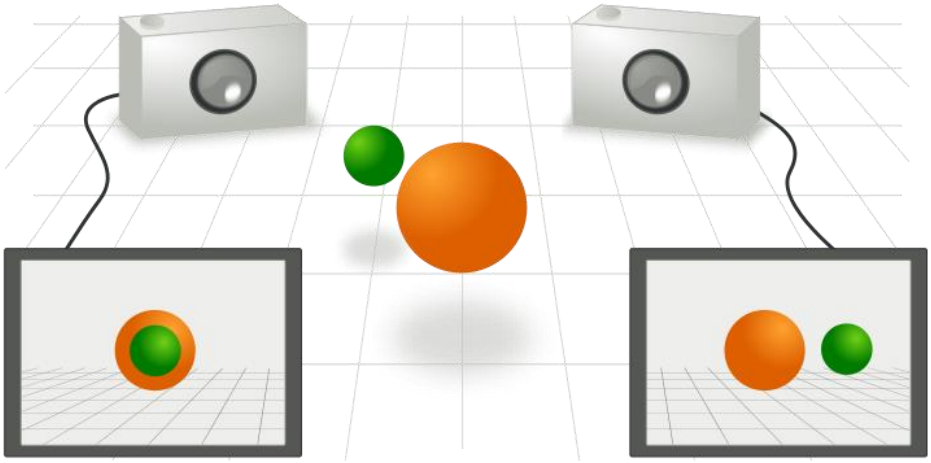


Figura 16 Diferentes perspectivas según observador

Un plano epipolar se define como aquel que contiene el punto del mundo tridimensional P y los dos ejes ópticos O_L,O_R, Fig. 17. También se definen las líneas epipolares como aquellas que surgen de la intersección entre el plano de la imagen y el plano epipolar. Estas líneas epipolares se representan en la imagen como líneas horizontales, ya que los ejes ópticos son paralelos entre si lo que lleva a que la restricción epipolar(En la práctica hay que rectificar):

$$v_1 = v_2 \tag{3.1.10}$$

Al cumplirse esta restricción el problema del emparejamiento de los puntos de las dos imágenes se da solo en una dimensión, en concreto el eje horizontal.

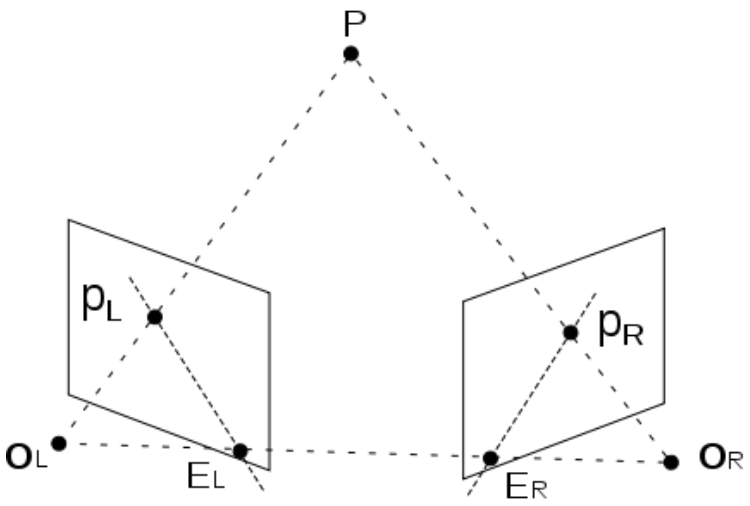


Figura 17 Plano epipolar

3.2.Mapas de disparidad

3.2.1. Introducción

La disparidad binocular es la pequeña diferencia entre los dos puntos de vista de un sistema de visión estéreo, gracias a esta pequeña diferencia debida a la posición de las camaras hace que podamos reconstruir imágenes en tres dimensiones a partir de 2 imágenes.

El mapa de disparidad que se construye a partir de ellas es una imagen en nivel de gris, que nos indica según la intensidad del pixel la profundidad del punto en el espacio. El cálculo de la disparidad en visión por computador es medir la distancia entre un punto de la imagen derecha y un punto de la imagen izquierda.

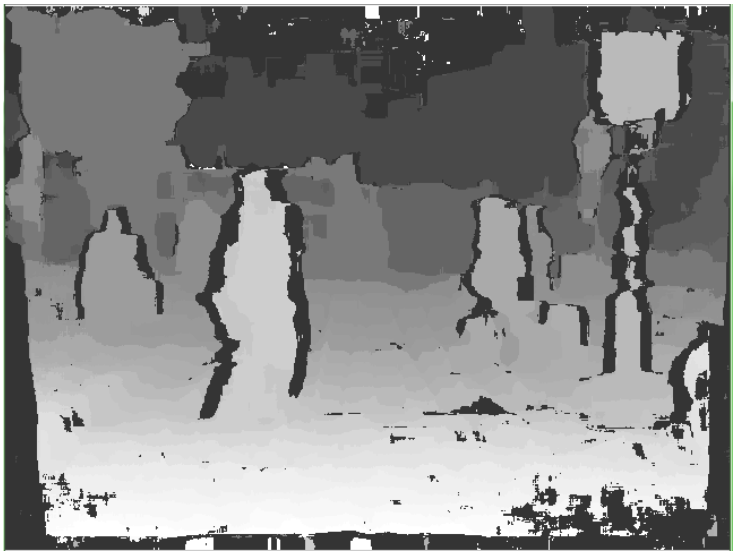


Figura 18 Mapa de disparidad equalizado

En la Fig. 18 podemos ver un mapa de disparidad ecualizado en el que se distinguen los diferentes niveles de gris. Según el nivel de gris que contenga el pixel, el objeto estará más cerca del objetivo o más lejos. Para poder calcular el mapa de disparidad se necesitan una serie de requisitos para que el coste computacional no sea alto. Una de estos requisitos es que las texturas de las imágenes no cambien según el punto de vista del observador, las imágenes se consideraran rectificadas y cumplen la restricción epipolar en la que la coordenada v de la imagen es la misma para las dos.

3.2.2. Algoritmos de cálculo.

Como se asume que los planos son paralelos entre las imágenes y además son paralelos a la dirección del desplazamiento entre frames. El siguiente paso es imponer las restricciones físicas y geométricas para obtener una relación razonable entre los objetos que se encuentra en las imágenes, las restricciones más utilizadas son:

- Restricción epipolar, de la cual ya ha sido mencionada anteriormente como la condición por el cual la coordenada *v* es igual para las dos imágenes.

- Restricción de orden, si la proyección de un objeto D esta la izquierda del objeto P en una imagen, ese objeto D también tiene que estar a la izquierda del objeto P en la otra imagen. Donde P es el objeto que tiene las mismas coordenadas para las dos imágenes.

- Restricción de unicidad, los puntos de la imagen izquierda o derecha solo pueden tener un solo punto de correspondencia en la imagen contraria.

- Restricción de semejanza, las características de los puntos emparejados deben ser similares (intensidad, color, etc).

Un factor muy importante a tener en cuenta son las oclusiones, estos puntos de oclusión son los puntos que se observan solo desde una imagen. Todos estos puntos se clasificaran con disparidad 0 igual que todos los puntos más alejados del objetivo.

El cálculo del mapa denso de disparidad debe plantearse en cuatro fases:

- Cálculo de la función de coste
- Agregación del coste
- Cálculo de la disparidad

- Post-procesamiento

3.2.2.1. Cálculo de la función de coste

El cálculo del nivel de semejanza se puede realizar mediante la función de coste, aunque hay una gran cantidad de métodos, los dos más usados son:

- Diferencia absolutas de intensidad denominada AD por su acrónimo en inglés (Absolute Intensisty Differences). Como el propio nombre indica se calcula la diferencia absoluta entre los valores de intensidad de cada imagen.

- Diferencias cuadráticas de intensidad o SD (Squared Intensisty Differences). Este método usa la diferencia cuadrática entre intensidades.

En ambos casos la menor diferencia será la que empareje los pixeles de cada imagen, también existen otros métodos como el normalized cross-correlation (correlación cruzada normalizada). Se basa en la comparación entre las covarianzas que miden la semejanza lineal entre puntos.

3.2.2.2. Agregación del coste

Se eligen regiones cercanas al pixel y se calculán sus diferencias de intensidad con los pixeles de esta ventana, podemos elegir distintas formas y tamaños de ventana. En la comparación de intensidades se tendrán en cuenta los costes promediados o sumados calculados en la función de costes.

Las ventanas más comunes son la ventana rectangular, estas ventanas son las más sencillas de todas. El algoritmo que se usa en este trabajo utiliza es una ventana cuadrada de un tamaño definido previamente. El centro de la ventana será el pixel de interés y vamos colocando los costes en la matriz rectangular.

Otra opción es la utilización de las ventanas adaptativas que usan tamaños formas y disposiciones del pixel de interés diferentes, para finalmente usar la ventana que menor coste haya tenido.

También se pueden utilizar otro tipo de métodos, como los pesos adaptativos. En el que se otorga más peso a los pixeles que tengan una mayor probabilidad de corresponder con el pixel de interés. Esta probabilidad se basa en la distancia que separa los pixels, la diferencia de color o de intensidad.

Otro método muy usado también es la difusión iterativa. En la que se difunde el valor de la función de coste a los pixeles vecinos al de interés. Esto se realiza durante varias iteraciones hasta que los valores cumple una condición puesta al inicio del programa, sino se impone una condición de parada el algoritmo continuara ejecutándose hasta que la diferencia de coste sea 0. Con lo cual es esencial establecer una condición de parada, de otro modo el programa estaría eternamente calculando ya que nunca se llegaría al coste 0.

3.2.2.3. Cálculo de la disparidad

Para el cálculo de la disparidad se disponen de una serie de métodos:

- Métodos locales, este método pone hincapié en las etapas de correspondencia y agregación. El cálculo de la disparidad se resuelve sencillamente. La posición que tenga un coste menor es de la que se extrae el valor. Un problema que surge con estos algoritmos es que solo hay una correspondencia por los que se pueden cometer errores. Por otro lado es sencillo de implementar en un programa informático.

- Las optimizaciones globales también son muy usadas, ya que en estos procesos la fase de agregación se suprime. Y se empiezan a utilizar fórmulas de minimización de la energía. Con este objetivo se buscara una ecuación que disminuya la energía global según la disparidad asociada a ese pixel.

Finalmente se busca los valores de d que minimizan la energía global y este será el valor que se usará.

Se están desarrollando una serie de métodos que buscan resolver ciertos problemas con la optimización global como el max-flow y graph-cut, esta implementación hace que los sistemas sean más eficientes.

Hay otros métodos como la programación dinámica, donde se optimizan las áreas de trabajo para localizar los mínimos globales. También existen los algoritmos cooperativos que utilizan sistemas no lineales cuyos resultados son similares a la optimización global.

Pero también existen nuevos algoritmos como Bobick e Intille (DP) y Kolmogorov y Zabih (GC) que introducen nuevos elementos como el coste de oclusión. Estos sistemas intentan eliminar la oclusión que surge alrededor de los objetos como se puede ver en las Fig. 20 y Fig. 21 .

- Bobick e Intille [11]: este algoritmo modela la oclusión y la integra en el cálculo del coste mínimo e introduce el concepto de Ground Control Points (GCP). Este algoritmo usa solo las scanlines, es decir solo considera en la comparación una única fila de pixeles y no las adyacentes.

Figura 19 Imagen del espacio de disparidad

Los GCPs son los puntos relevantes de la imagen del espacio de disparidad, Fig. 19, esto hace que los puntos del DSI que son GCP sean (x_G,d_G). Esto implica lo siguiente en las imágenes, $x_L= x_G$ y $x_R= x_G+d$ se corresponden. Lo que hace que las oclusiones alrededor de los objetos se ignoren. Pero si en este sistema aplicamos un coste demasiado bajo de oclusión las imágenes empiezan a distorsionarse, los resultados de este algoritmo pueden verse en la Fig. 20.

Figura 20 Mapas de disparidad obtenidos. De izquierda a derecha aumentando el coste de oclusión.

Podemos observar que al aumentar el coste de oclusión la imagen se va distorsionando en el plano horizontal. Mientras que si tenemos un coste de oclusión muy bajo, esto se reflejara en la imagen generando mucho ruido. Por eso es imprescindible llegar a un compromiso entre tener la imagen con una cierta cantidad de oclusión y una cierta distorsión de la imagen.

- Kolmogorov y Zabih [11], el algoritmo propuesto se basa en el método de corte de grafos e imponiendo una restricción de unicidad. Para la obtención eficiente del método de grafos se utilizan las siguientes ecuaciones:

$$E(L) = E_{data}(L) + E_{oclu}(L) + E_{smooth}(L) = \tag{3.2.1}$$

$$= \sum (I(p) - I(q))^2 + \sum C_p T(N_c = 0) + \sum_{(p,q)\in N} D_{p,q} T(L_p \neq L_q)$$

Donde L es la configuración de las posibles etiquetas a colocar en cada punto de grafo, E_{data} es la energía de los datos, E_{oclu} es la energía de la oclusión y E_{smooth} es la energía de suavizado. Para el cálculo de la disparidad las etiquetas serán los valores de disparidad. Este algoritmo al contrario que él anterior sí que compara con una área más grande y no solo con las scanlines. A la hora de identificar si los valores son oclusión utilizan el parámetro λ, con este parámetro se puede configurar las componentes de la energía, solo modificando λ. En la Fig. 21 se puede apreciar como aumentando el factor λ la oclusión va disminuyendo.

Figura 21 Mapas de disparidad obtenidos. De izquierda a derecha aumentando λ

Podemos observar que si se aumenta mucho el valor de λ se pierden detalles incluso objetos importantes como la lámpara.

Estos dos últimos métodos son muy útiles ya que uno de los mayores problemas es que al obtener el u-disparity los valores de oclusión tienen el mismo valor que los objetos en el infinito (d=0), esto hace que se cometan errores en el etiquetado.

3.2.3. Post procesamiento.

Finalmente el post-procesamiento ayuda a corregir pequeñas discontinuidades que se dan en las imágenes, refinando la imagen para tener un etiquetado mejor.

Este refinamiento puede llevarse a cabo en varias etapas y mediante diversos algoritmos. Estos pueden ser desde el gradiente iterativo a un ajuste a curvas de nivel de disparidad. Lo que se obtiene de aplicar estos algoritmos es un aumento de la resolución del mapa de disparidad, esto implica un aumento en el tiempo de cálculo. Sin embargo al tener una mejor resolución de la disparidad, tenemos una mejor información del objeto proyectado en la imagen. Por ultimo añadir que estos algoritmos solo pueden aplicarse a imágenes “suaves” es decir que tenga una disparidad constante y no tengan grandes saltos. Es decir las regiones sobre las que se aplica deben pertenecer a la misma superficie.

Existen otros métodos de post-procesamiento. Estos se basan en detectar las áreas ocluidas mediante el cross-checking. Esto consiste en comparar los mapas de la imagen

derecha con el de la izquierda, él de la imagen izquierda con él de la imagen derecha y marcando como ocluidos todos los pixels que tengan una disparidad distinta.

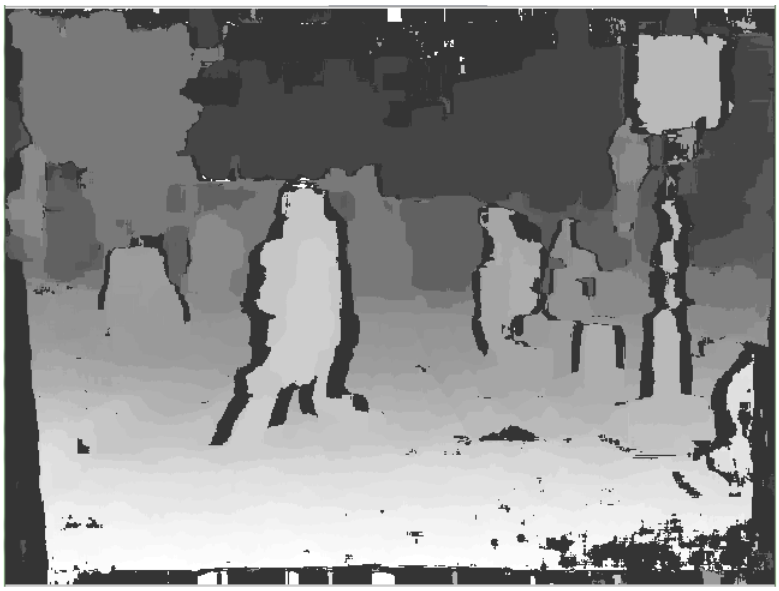
Finalmente se localizarán todas las zonas ocluidas o sin un valor de disparidad asociado (objetos muy lejanos). Con estos puntos localizados se realizara una interpolación con los pixeles adjuntos para asignar un valor de disparidad al pixel.

Otros post-procesamientos pueden utilizar filtros para la eliminación de ruidos pero que conserven los bordes de las imágenes, estos filtros pueden ser la mediana o filtros bilaterales para eliminar los errores de correspondencia y zonas ocluidas.

3.3.U-disparity

La imagen del u-disparity se genera a partir del mapa de disparidad, este contiene el histograma por columnas del mapa de disparidad. Los obstáculos que se encuentran perpendiculares al objetivo. Quedan representados como líneas horizontales y la intensidad representa la altura del objeto en relación con la posición del observador.

(A)



(B)

(C)

Figura 22 (A) Imagen de la cámara derecha. (B) Mapa de disparidad. (C)U-disparity

Como se puede observar en la Fig. 22 hay unos peatones cruzando la calle y estos quedan reflejados en el mapa de disparidad como dos objetos rodeados de oclusión. Finalmente podemos ver como en la posición horizontal del u-disparity aparecen unos objetos cercanos a la cámara y con una alta intensidad como se puede ver en la Fig. 23.

Figura 23 Peatones en el u-disparity

3.4. Descriptor LBP

El descriptor LBP (Local Binary Pattern) es el descriptor que relaciona mediante un byte los valores que se encuentran alrededor del pixel que estamos estudiando, Fig. 24.

Este descriptor funciona de la siguiente manera, seleccionamos una característica que tenga ese pixel, esta puede ser color, nivel de intensidad, posición, etc. Una vez

fijada la característica elegida se coge el valor de esta y se compara con los 8 pixeles alrededor del pixel en estudio.

22	33	38
43	48	32
50	60	55

Figura 24 Pixeles usados para el LBP

Con estos datos se realiza la comparación con el valor central y si este es mayor, el valor del pixel exterior será cero y si el valor central es menor el pixel tomara el valor 1, Fig. 25.

0	0	0
0	key	0
1	1	1

Figura 25 Codificación de los valores de la Figura 25

Los valores obtenidos se codificaran en un byte, en este proyecto se codificara de la siguiente manera, pero se podría colocar los bit en el orden que se quiera.

Bit 0	Bit 1	Bit 2
Bit 7	Key	Bit 3
Bit 6	Bit 5	Bit 4

Tabla 1 Codificación bits LBP

La codificación que se observa en la Tabla 1, cada bit corresponde con un bit del byte donde se quedara almacenado el valor del LBP, donde el bit 0 es el bit menos significativo y el Bit 7 el más significativo.

Este descriptor los que nos ofrece es la información del entorno en el que se encuentra el pixel que estamos analizando, esto es muy útil cuando necesitas la información del contexto en el que se está trabajando.

3.5.Redes Neuronales [16]

3.5.1. Introducción

Las redes neuronales artificiales son sistemas de aprendizaje y procesamiento automático, esta inspirados en la forma en la que funciona el sistema nervioso animal. Las redes neuronales suelen constar de 3 capas, la capa de entrada, la capa oculta que es donde se encuentran las neuronas y la capa de salida, Fig. 26.

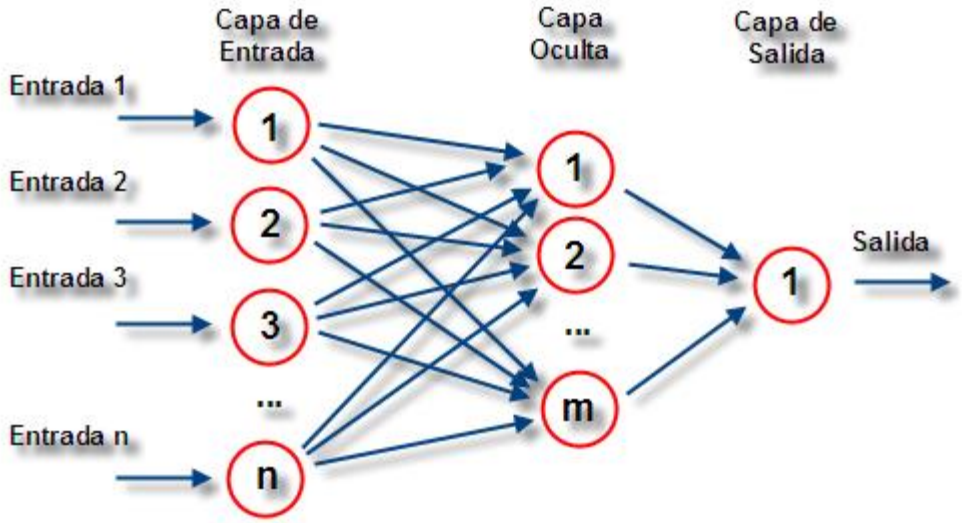


Figura 26 Esquema de red neuronal.

Las redes neuronales surgieron por primera vez en 1943, pero no fue hasta mediados de los 80 cuando realmente surgieron con más fuerza este tipo de sistemas de inteligencia artificial. Las redes neuronales en la actualidad se usan en una gran cantidad de campos, tanto científicos como comerciales, algunos ejemplos son la clasificación de proteínas, o la inteligencia artificial en ciertos videojuegos, en las que los personajes controlados por el ordenador son capaces de aprender del jugador sus tácticas. También se usan en el ámbito industrial para la clasificación de productos defectuosos mediante el aprendizaje de las formas correctas de los productos.

Las redes neuronales tienen que realizar un entrenamiento. Este entrenamiento o aprendizaje consiste en dar a la red un número de muestras, con una serie de características, esta características son aquellas que se introducen en la capa de entrada y son las que la red neuronal usara cuando el entrenamiento acabe para identificar los patrones. Junto con estas muestras de entrada también se introducen los datos de salida, es decir le proporcionamos a la red todas la información que necesita para comparar los datos de entrada con los datos de salida. Con estos datos se realiza el entrenamiento que consta de tres fases:

- Aprendizaje: se destinan una serie de datos al aprendizaje, aquí la capa de oculta de neuronas compara las muestras de entrada con las de salida y cada neurona va creando sus relaciones entre estos datos.

- Validación: la validación se realiza cada cierto número de iteraciones, aquí la red neuronal va comprobando si la capa oculta ha terminado su aprendizaje. Esto lo hace comparando el resultado de introducir un dato en la capa de entrada, realizar el cálculo y finalmente comparar ese resultado con el resultado introducido con las muestras. En caso de que no se acierte la red se recalcularía introduciendo esos datos. Una vez terminada toda la validación que se suele realizar con un 15% de los datos introducidos se pasa a la fase de test.

- Test: en esta fase se comparan los datos calculados por la red neuronal y los datos introducidos, pero no se vuelve a recalcular la red. De esta fase saldrán los datos de porcentaje de acierto. Aquí también es usual el 15% de los datos de muestra.

3.5.2. Fundamentos matemáticos

Uno de los modelos más exitosos de redes neuronales son las denominadas feed-forward neural networks (red neuronal feed-forward). Estas redes también son conocidas como redes multicapa. Estas redes al fundamentarse sobre modelos de

regresión lógica, utilizan no-linealidades continuas. Hace que este modelo sea más compacto y más rápido de evaluar que otros modelos basados en las no-linealidades discontinuas. Pero en contra surgen algunos problemas debidos a que la función de aproximación al patrón no es una función convexa, esto hace que sea mejor invertir más tiempo en el entrenamiento, para obtener un buen modelo y aprovechar el modelo compacto que ofrecen las redes neuronales.

Los modelos feed-forward están basados en combinaciones lineales de funciones no lineales $\phi_j(x)$ en la siguiente forma:

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)) \tag{3.5.1}$$

Donde f() es una función no lineal de activación. El objetivo de la red neuronal es obtener las funciones $\phi_j(x)$ que deberán depender de los parámetros a introducir. Finalmente estos parámetros serán ajustados por los pesos w_j durante el entrenamiento. Las redes neuronales usan las funciones de tipo (3.5.1) como función básica.

Estas nuevas funciones compiten con las funciones clásicas de las redes neuronales que utilizan modelos basados en transformaciones funcionales. Primero se construyen *M* combinaciones lineales de las variables internas x_i con la forma:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{3.5.2}$$

Donde el índice *j=1,…,M* y el superíndice (1) indican los parámetros de la primera capa neuronal, los parámetros w son los pesos ponderados de las ecuaciones que se ajustaran según el entrenamiento. Las cantidades a_j son conocidas como las activaciones y todas ellas son transformadas usando una función de activación diferenciable y no linear *h()* para obtener.

$$z_j = h(a_j) \tag{3.5.3}$$

Estas cantidades corresponden con los resultados de las funciones básicas (3.5.1) y se denominan unidades o neuronas ocultas. Las funciones no lineales *h()* suelen ser de tipo sigmoidal como funciones lógicas sigmoidales o la tangente hiperbólica. Utilizando la ecuación (3.5.1) los valores obtenidos son vueltos a combinar linealmente para obtener los output unit activation.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \qquad (3.5.4)$$

Donde *k=1,…,K*, y K es el número total de outputs. Esta transformación corresponde con la segunda capa de la red. Finalmente estos valores serán transformados utilizando una función de activación de salida. La elección de esta función estará determinada por la naturaleza de los datos de salida. En la Fig. 27 se puede observar las diferentes partes de una red neuronal, tanto como los inputs como los outputs y las capas ocultas que se encuentran entre estas. Se pueden observar las distintas conexiones entre las diferentes neuronas, dando lugar a la red capaz de identificar patrones.

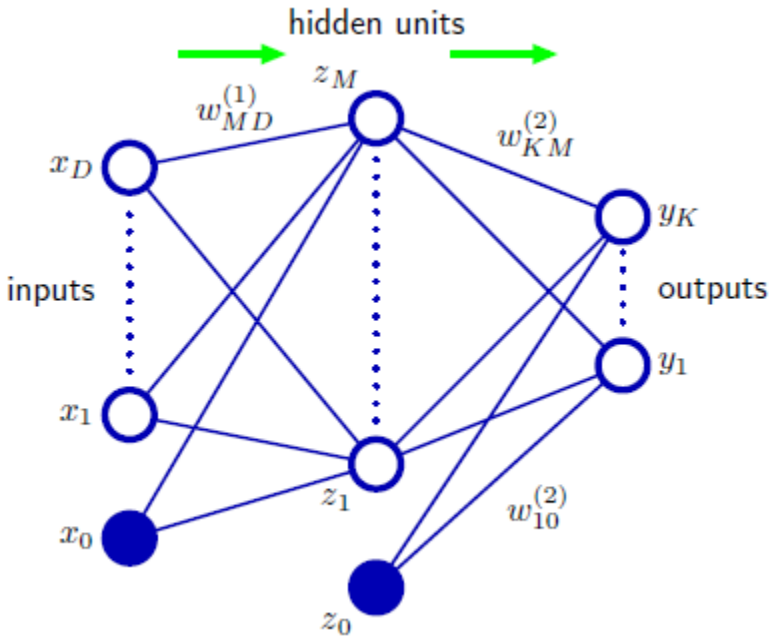


Figura 27 Red neuronal y sus capas

Para problemas de regresión estándar la función de activación es la identidad $y_k=a_k$. Pero para clasificación binaria múltiple se deben utilizar funciones sigmoidales como

$$y_k = \sigma(a_k) \tag{3.5.5}$$

Donde la función σ es

$$\sigma(a) = \frac{1}{1+\exp(-a)} \tag{3.5.6}$$

Finalmente si se combinan todas las etapas del proceso de creación de una red neuronal la ecuación resultante será:

$$y_k = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)}) \tag{3.5.7}$$

Los parámetros de tendencia se unifican con los pesos para formar el vector w. Así el modelo de red neuronal es una función simple no-linear que parte de un conjunto de inputs (x) y outputs (y).

Esta función puede representarse como un diagrama de red como podemos observar en la Fig. 27, estas redes no proporcionan un valor probabilístico sino determinista ya que cada nodo representa una variable totalmente determinista y no estocástica.

Para simplificar la ecuación de la red neuronal podemos incluir los parámetros de tendencia dentro de los pesos, de manera que la ecuación será:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i \tag{3.5.8}$$

Si esto lo aplicamos a la función final.

$$y_k = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i)) \qquad (3.5.9)$$

Si en lugar de funciones de activación no-lineales tomamos funciones lineales, entonces siempre se podrá encontrar una red sin capas ocultas. Esto parte de que una composición de funciones lineales es una función lineal. Sin embargo si el número de neuronas ocultas es bajo, y si este es menor que el número de inputs y outputs, las funciones que genera la red neuronal no son las más generalistas que se podrían obtener, de manera que se pierde información debido a la pérdida de dimensión en la capa oculta.

3.5.2.1. Espacio de pesos simétricos

Otra propiedad de estas redes neuronales feed-forward es el espacio de pesos simétricos. Con una red neuronal de dos capas con M neuronas ocultas y utilizando funciones tangenciales hiperbólicas. Si se cambia el signo de todos los pesos y de todos los parámetros de tendencia dentro de una neurona oculta (vector w), entonces para los datos de entrada el signo de activación será invertido, obteniendo una función impar, $\tanh(-a) = -\tanh(a)$. Este signo puede verse compensado cambiando todos los signos de los pesos que tiene que ver con esa neurona. Este cambio de signos no supone ningún cambio en nuestra red global, con lo que hemos obtenido 2 vectores de peso que ofrecen los mismos resultados. Así que para *M* neuronas ocultas habrá *M* cambios de signo simétricos con lo que nos da 2^M vectores equivalentes.

Equivalentemente si se cambian los valores del vector w que tienen que ver con las salidas y entradas de una neurona oculta. Llegamos a la misma conclusión de que la red neuronal no ha sido alterada. En este caso se obtendría que para *M* neuronas tendríamos *M!* vectores de pesos, si esto se combina en los anteriormente expuesto obtenemos el resultado de $M! \cdot 2^M$ vectores de pesos. Esto quiere decir que para un mismo problema hay una cantidad inmensa de resultados y redes neuronales.

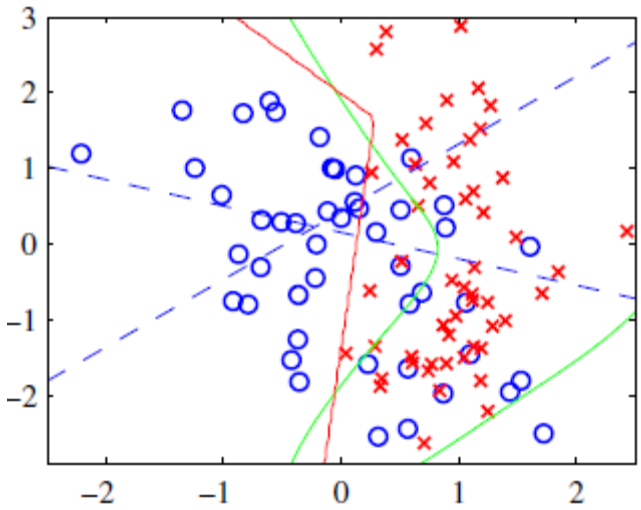


Figura 28 Soluciones Red Neuronal

En la Fig. 28 podemos observar distintas soluciones para un caso simple de dos entradas, dos capas y una salida. Así podemos ver que hay varias soluciones para los valores dados. La solución de la líneas de puntos azules en la que se utiliza un z=0.5 para cada neurona, y la línea roja ofrece una solución de y=0.5. Podemos observar que las líneas rojas se acercan más a la solución ya que el factor y es el factor después de la primera capa de neuronas ocultas y el aprendizaje se realiza en esa capa, con lo que solo estamos realizando un pequeños ajuste al aprendizaje de la red. Por último decir que la solución óptima son las líneas verdes.

4. Desarrollo del proyecto

El objetivo del proyecto es realizar una red neuronal que sea capaz de diferenciar entre 6 clases de objetos en una imagen. Estos objetos serán:

- Oclusión
- Calzada
- Vehículo
- Peatones
- Cielo
- Obstáculos (Objetos estáticos, edificios, farolas, señales)

Para obtener la información se usan imágenes estéreo, de la cuales se obtiene su u_disparity y este es etiquetado. Con el u_disparity etiquetado se obtendrán una serie de características, que se introducirán en la red neuronal. Esta red será la que se encargue finalmente de etiquetar las imágenes que deseemos.

Aunque la oclusión no se observara en la imagen es necesario etiquetarla para poder identificarla en el u_disparity y así poder diferenciarla de los demás objetos en la imagen.

Este proyecto se podría dividir en tres partes, preparación de datos, creación de la red neuronal y por último programa de análisis de imágenes.

4.1.Preparación de datos

En esta fase se preparan los datos para que se puedan introducir en la red neuronal, ya que Matlab exige que los datos a introducir en la red neuronal sean matrices. Que serán los datos que una vez terminada y probada la red se introducirán en el programa, que usa la red para etiquetar las imágenes. Los otros datos que necesita son los *targets* o datos ya etiquetados, con estos datos la red se entrenara para ser capaz de identificar los objetos. Todos estos datos deben de ser introducidos

en forma de matriz, donde las columnas serán el número de muestras y las filas serán las características de las muestras. Y los datos de los target tiene que estar en la misma posición de la matriz, es decir el pixel que corresponde a la columna i en la matriz de inputs, el target (etiqueta) de ese pixel tiene que estar en la columna i de la matriz de targets.

Las etiquetas que se utiliza en el u_disparity para identificar los objetos serán números del 0 al 5:

- Oclusión = 0
- Calzada = 1
- Vehículo = 2
- Peatones = 3
- Cielo = 4
- Obstáculo = 5

A la hora de realizar el etiquetado se tendrán dos opciones, el etiquetado de imágenes completas y el etiquetado por partes de la imágenes. Se realizan estas dos maneras de etiquetado debido a que en los u_disparity hay una gran cantidad de puntos que pertenecen a la calzada y a la oclusión, lo que produce una sobrepoblación de muestras de estas dos categorías, haciendo que el entrenamiento de la red neuronal no sea efectivo, ya que es posible que no se tengan muestras de otros objetos como peatones o vehículos, cuyas muestras son mucho menos abundantes, pero igual de importantes.

4.1.1. Etiquetado de imágenes completas

Para la realización de este apartado se ha creado una función en Matlab, este función se denomina etiquetar.m. Donde los datos de entrada son el mapa de disparidad de la imagen, Fig. 29 (B), y una de las imágenes de la cámara estéreo, Fig. 29 (A). La salida es otro archivo .tif en el que se representa el u_disparity, Fig. 29 (C), pero en lugar de encontrar los valores de disparidad, lo que encontraremos serán los datos del etiquetado, es decir los pixeles de esta imagen tendrán valores de gris entre 0 y 5.

Para realizar estos introduciremos en la consola de Matlab el comando,

etiquetar('nombre mapa disparidad.tif','nombre imagen.tif')

Una vez ejecutado este comando se mostraran por pantalla las tres imágenes de la siguiente manera:

(A)

(B)

(C)

Figura 29 (A) Imagen derecha. (B)Mapa de disparidad ecualizado. (C) U_disparity

Al mismo tiempo surgirá el siguiente menú en la consola de Matlab:

```
Selecione objeto a etiquetar: 1-peaton,2-vehiculo,3-cielo,4-calzada y obstaculos,0-Fin
```

Donde introduciremos los valores de 1, 2, 3, 4 ó 0, estos valores seleccionaran la etiqueta a colocar, en el área que seleccionaremos con la función de Matlab *getrect,* Fig. 30. Esta función devuelve el tamaño de un rectángulo, seleccionado con el ratón sobre el u_disparity, esta área será etiquetada con la selección realizada anteriormente. La función solo permite seleccionar con rectángulos si se selecciona los valores 1, 2 ó 3. El valor 4 deberá seleccionarse en último lugar, y una vez ya se haya etiquetado los demás objetos. Ya que con este valor todos los pixeles no etiquetados de la imagen serán etiquetados como obstáculo, calzada u oclusión.

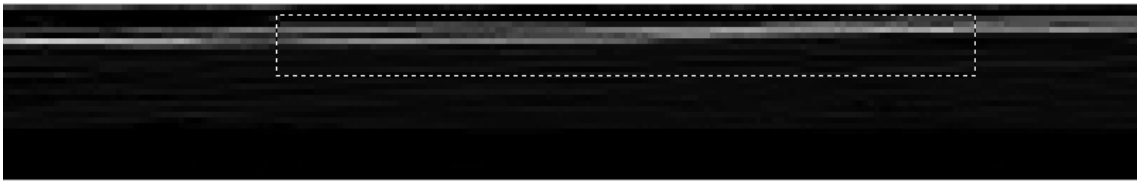


Figura 30 Ampliación del u_disparity con detalle de la función getrect

Los criterios de etiquetado para el área seleccionada o para la ejecución del comando 4 es el siguiente:

- Si el valor de disparidad para ese pixel es 0 la etiqueta asignada será 0 (oclusión).

- Si se ha elegido las opciones 1, 2 ó 3 cualquier valor de disparidad mayor que 25 recibirá su etiqueta correspondiente es decir para 1 recibirá un 3 de peatón, con el 2 recibirá un 2 de vehículo y con un tres recibirá un 4 de cielo. Se elige el valor 25 porque significa que hay más de 25 pixeles con ese valor de disparidad, lo que implica que hay un objeto a tener en cuenta, ya que tiene 25 o más pixeles de altura.

- En caso de seleccionar la opción 4 si la disparidad es mayor que 25 recibirá la etiqueta de obstáculo (todo lo que no éste etiquetado anteriormente pasar a ser un obstáculo), si el valor es mayor que 0 y menor que 25 el pixel recibirá la etiqueta de calzada.

Una vez etiquetada toda la imagen (se ha pulsado la opción 4), se introducirá el valor 0 en el menú y aparcera en la pantalla el texto *Guardar como* y se introducirá el nombre entre comas y con la extensión .tif, ej: 'Nombre etiquetas.tif'.

4.1.2. Etiquetado parcial de imágenes

Debido a que en las imágenes hay un predominio de los pixeles asociados a la oclusión y a la calzada, para poder introducir más datos sin agregar más imágenes completas, que solo aportarían más pixeles de calzada y oclusión. Solo se introducirán los datos de las áreas seleccionadas por el usuario.

Para poder realizar esto se utiliza un programa basado en el programa anterior pero con ligeras modificaciones. Estas modificaciones permiten que solo se etiqueten las áreas adecuadas, evitando introducir muchos datos de calzada.

En esta versión del programa anterior se creara una matriz del mismo tamaño que el u_disparity que se va a etiquetar. Esta matriz se inicializara con el valor número 6, está etiqueta nos servirá para discriminar estos pixeles a la hora de la preparación de los datos.

Una vez obtenida la matriz de seises, comenzaremos a usar el programa igual que la versión completa, es decir tendremos que elegir qué tipo de objeto queremos etiquetar y después lo seleccionaremos sobre el u_disparity con el ratón. Esto hace que en el área seleccionada, todos los pixeles de la matriz se etiqueten con algún valor de 0 a 5. En este programa si seleccionamos el número 4, este no etiquetara el resto de la imagen sino que funcionara como los valores 1,2 ó 3, haciendo que solo se

etiqueten con el valor 4 los objetos que tengan un valor de disparidad mayor que 25, y que se encuentren en el área selecionada.

Hay que tener en cuenta que solo se podrá etiquetar un pixel si este tiene el valor número 6 en la matriz creada. Esto quiere decir que lo primero que se etiquete es lo que perdurara, no hay sobrescritura.

Finalmente obtendremos una matriz con zonas etiquetadas, y todas aquellas zonas que no se seleccionaran permanecerán con el valor 6 para descartarlos a en el aprendizaje. El último paso es darle el nombre al archivo .tif que se guardará.

4.1.3. Depuración de imágenes completas

Debido a ruidos en las imágenes es necesario implementar un programa de depuración para evitar datos falsos. Como no podemos saber qué tamaño tienen los objetos, la depuración se centra en eliminar el ruido de las zonas de oclusión.

En las zonas de oclusión no puede haber puntos etiquetados, ya que estos están tapados por el objeto que tenemos delante. Como podemos ver en la Fig. 31 tenemos un u_disparity etiquetado y un u_disparity depurado.

(A)

(B)

Figura 31 (A) U_disparity sin depurar (B) U_disparity depurado

En el depurado de imágenes como se puede observar en la figura anterior se eliminan de la oclusión todos los datos que no son correctos.

4.1.4. Tratamiento de datos

En este apartado se explicara cómo se deben organizar los datos, para poder introducirlos en la red neuronal. En la red neuronal se introducirán dos matrices que tendrán la siguiente estructura.

- El número de columnas tendrá que ser igual en ambas matrices y este número será la suma de todos los pixeles que se utilizan como muestras.

- El número de filas viene dado en ambas matrices como las características de cada pixel, en el caso de los datos de entrada, y en los datos de salida el número de filas es el número de etiquetas.

El tratamiento para los objetivos es muy sencillo e igual para los distintos tipos de entradas, se utilizaran dos tipos de características para las entradas. En lo que respecta a los objetivos se creara una matriz de 6 filas y X columnas, donde X es el número de muestras.

Las filas numeradas de 1 a 6, ya que Matlab por defecto numera desde el 1 y no desde el 0, corresponde con la numeración de la etiqueta. La etiqueta 0 corresponde

con la primera fila, el 1 con la segunda y así sucesivamente. Se utiliza un sistema binario para los objetivos, con el que la matriz se inicializa toda a cero y solo se colocara un 1 en la fila que corresponda, según la etiqueta que lea del u_disparity etiquetado. El programa lee directamente del u_disparity anteriormente etiquetado.

En el caso de los datos de entrada se tendrán en un caso 4 características por pixel, y en el segundo 11 características por pixel. Estos quiere decir que el primer caso tendremos una matriz de 4 filas y X columnas, esta X es la misma para los objetivos y el segundo caso, en este último tendremos una matriz de 11 filas y X columnas. Los datos de entrada se obtienen del u_disparity. Las tres primeras características son comunes para los dos casos:

- Primera fila: en esta fila se introducirán los valores de la coordenada *X*, esta es la posición en el eje U del pixel tanto en el u_disparity como en la imagen visible.

- Segunda fila: aquí se colocaran los valores de la coordenada *Y* del u_disparity, este valor es el valor de disparidad en el mapa de disparidad de la imagen.

- Tercera fila: el valor introducido en esta fila es el valor de la intensidad que tiene el pixel en el u_disparity, que corresponde para el caso de los obstáculos, a su altura medida en pixeles.

Después de estas tres filas comunes a los dos casos de datos a introducir, tenemos las siguientes opciones.

4.1.4.1. LBP compacto

En el LBP compacto [10] solo se utiliza una fila de la matriz para introducir el valor del LBP del pixel muestra. Para esto se calculara el valor del LBP en los 8 vecinos del pixel dándole los valores de la tabla 1, presente en el apartado 3.4. De esta manera el valor del bit 0 es el menos significativo y el valor más significativo es el bit 7.

Utilizando esta manera de introducir el valor del LBP, obtenemos de una manera compacta la información del entorno del pixel, de esta manera se pierde la información de direccionalidad. Pero la cantidad de cálculos en la red neuronal se ve reducida al tener muchas menos características de entrada.

Este dato va desde 0 a 255, donde los valores altos significan que el pixel está rodeado de pixeles con un valor de intensidad menor, y si el valor del LBP es pequeño significa que el pixel está rodeado de pixeles con una menor intensidad.

El LBP se introducirá en la cuarta fila de la matriz de características.

4.1.4.2. LBP extendido

En este caso el descriptor LBP se introducirá de manera individual, cada bit sera una fila de la matriz de características, esto hace que la información del entorno este constituida por 8 características, en la práctica lo que significa es que cada pixel representa una dirección, Tabla 2.

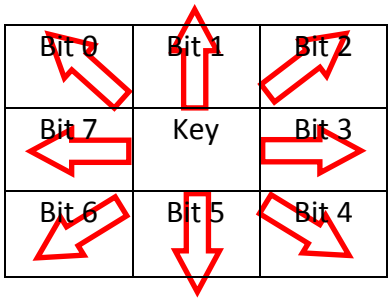


Figura 32 Direcciones de cada bit

Al introducir cada pixel por separada dentro de la matriz, lo que se consigue es darle a la red neuronal un valor del entorno y además el valor de la dirección de ese entorno. Usando el LBP compacto se pierde información de direccionalidad en el entorno al pixel, ya que solo obtenemos los valores de alrededor. El LBP compacto otorga una mayor valor a los bits más significativos, con lo que se pierde mucha información del entorno de los bits 0 a 2, ya que tienen muy poca influencia.

Utilizando el LBP extendido los datos se colocan desde la fila 4 a la fila 11, los pixeles se colocaran empezando con el bit 0 en la fila 4, el bit 1 en la fila 5 y así sucesivamente hasta llegar al bit 7 en la fila 11.

4.1.4.3. Tratamiento de imágenes completas

Para el tratamiento de imágenes completas, se introducirá en Matlab los datos necesarios para ello, que en este caso será el mapa de disparidad de la imagen, del que se calculará su u_disparity y a continuación se introducirá el archivo .tif de las etiquetas, que hemos obtenido en el proceso de etiquetado manual.

El programa recorrerá el u_disparity original y el etiquetado en el mismo orden, e ira obteniendo los valores de las características del u_disparity original y los objetivos del u_disparity etiquetado. Es imprescindible que las matrices se recorran en el mismo orden, ya que la red neuronal asocia cada columna a una muestra, es decir en la red neuronal la columna 4 en las características corresponde con la columna 4 en la matriz de objetivos.

Para hacer la introducción de datos en la red neuronal más sencilla, en el programa de etiquetado se podrán introducir tantas imágenes como uno quiera, de esta manera se obtienen todos los datos en dos matrices, que se podrán introducir directamente en la red neuronal. Esto hace que el tratamiento de datos sea mucha más sencillo.

```
>> tratamientoLBP
introduzca mapa de disparidad
'cross1.tif'
introduzca etiquetas
'eti1.tif'
Continuar pulse 1. Terminar pulse 0
```

Figura 33 Captura del programa de tratamiento LBP

Como se puede ver en la Fig. 33, el programa pide que se pulse 1 si se quiere continuar introduciendo imágenes para las muestras, o si se quiere terminar

simplemente se pulsa 0 y el programa devolverá la matriz de inputs y la matriz de características.

4.1.4.4. Tratamiento de imágenes parciales

El etiquetado parcial de imágenes ofrece unas imágenes distintas, ya que contienen una 7ª etiqueta, con el valor número 6 y que significa pixel no etiquetado. El programa para este caso funciona exactamente igual que el anterior, pero cuando lee del u_disparity etiquetado el valor 6, el programa se saltara ese pixel y no se incluirá en las matrices de características y de objetivos.

Esto producirá en matrices mucho más pequeñas que las matrices de imágenes completas.

4.1.4.5. Esquema de las matrices

El esquema para las matrices es el siguiente:

Matriz de características (LBP Compacto)

$$\begin{bmatrix} PosX_1 & \cdots & PosX_n \\ Disparidad_1 & \vdots & Disparidad_n \\ Valor\ u\_disparity_1 & \vdots & Valor\ u\_disparity_n \\ LBP_1 & \cdots & LBP_n \end{bmatrix}$$

Matriz de características (LBP Extendido)

$$\begin{bmatrix} PosX_1 & \cdots & PosX_n \\ Disparidad_1 & \vdots & Disparidad_n \\ Valor\ u\_disparity_1 & \vdots & Valor\ u\_disparity_n \\ Bit\ 0\ LBP_1 & \vdots & Bit\ 0\ LBP_n \\ Bit\ 1\ LBP_1 & \vdots & Bit\ 1\ LBP_n \\ Bit\ 2\ LBP_1 & \vdots & Bit\ 2\ LBP_n \\ Bit\ 3\ LBP_1 & \vdots & Bit\ 3\ LBP_n \\ Bit\ 4\ LBP_1 & \vdots & Bit\ 4\ LBP_n \\ Bit\ 5\ LBP_1 & \vdots & Bit\ 5\ LBP_n \\ Bit\ 6\ LBP_1 & \vdots & Bit\ 6\ LBP_n \\ Bit\ 7\ LBP_1 & \cdots & Bit\ 7\ LBP_n \end{bmatrix}$$

Matriz de objetivos

$$\begin{bmatrix} Etiqueta\ oclusión_1 & \cdots & Etiqueta\ oclusión_n \\ Etiqueta\ calzada_1 & \vdots & Etiqueta\ calzada_n \\ Etiqueta\ vehículo_1 & \vdots & Etiqueta\ vehículo_n \\ Etiqueta\ peatón_1 & \vdots & Etiqueta\ peatón_n \\ Etiqueta\ cielo_1 & \vdots & Etiqueta\ cielo_n \\ Etiqueta\ obstaculo_1 & \cdots & Etiqueta\ obstaculo_n \end{bmatrix}$$

Debido al exceso de datos de oclusión se eliminan la mitad de las muestras de estos dos grupos, para tener una cantidad de muestras que este más equilibrada, en lo que concierne al tipo de muestras.

Finalmente se concatenaran las matrices, las matrices de características por un lado y las matrices de objetivos por otro. De esta manera ya tendremos lista las matrices de inputs y target necesarias para la red neuronal.

Matrices usadas en la red neuronal

Inputs

$$\begin{bmatrix} PosX_1 & \cdots & PosX_n & PosX_{n+1} & \cdots & PosX_{n+m} \\ Disparidad_1 & \vdots & Disparidad_n & Disparidad_{n+1} & \vdots & Disparidad_{n+m} \\ Valor\ u\_disparity_1 & \vdots & Valor\ u\_disparity_n & Valor\ u\_disparity_{n+1} & \vdots & Valor\ u\_disparity_{n+m} \\ LBP_1 & \cdots & LBP_n & LBP_{n+1} & \cdots & LBP_{n+m} \end{bmatrix}$$

Targets

$$\begin{bmatrix} Etiqueta\ oclusión_1 & \cdots & Etiqueta\ oclusión_n & Etiqueta\ oclusión_{n+1} & \cdots & Etiqueta\ oclusión_{n+m} \\ Etiqueta\ calzada_1 & \vdots & Etiqueta\ calzada_n & Etiqueta\ calzada_{n+1} & \vdots & Etiqueta\ calzada_{n+m} \\ Etiqueta\ vehículo_1 & \vdots & Etiqueta\ vehículo_n & Etiqueta\ vehículo_{n+1} & \vdots & Etiqueta\ vehículo_{n+m} \\ Etiqueta\ peatón_1 & \vdots & Etiqueta\ peatón_n & Etiqueta\ peatón_{n+1} & \vdots & Etiqueta\ peatón_{n+m} \\ Etiqueta\ cielo_1 & \vdots & Etiqueta\ cielo_n & Etiqueta\ cielo_{n+1} & \vdots & Etiqueta\ cielo_{n+m} \\ Etiqueta\ obstaculo_1 & \cdots & Etiqueta\ obstaculo_n & Etiqueta\ obstaculo_{n+1} & \cdots & Etiqueta\ obstaculo_{n+m} \end{bmatrix}$$

4.2.Entrenamiento red neuronal

Para el cálculo de la red neuronal se utilizara la herramienta de Matlab, el uso de esta herramienta es muy sencillo a la vez que intuitivo, lo único que se necesita es tener los datos de inputs y targets bien ordenados y preparados. Para inicializar esta herramienta simplemente se ejecutara el comando nnstart en la consola de Matlab. Al ejecutarlo nos aparecerá el siguiente menú, Fig. 34.

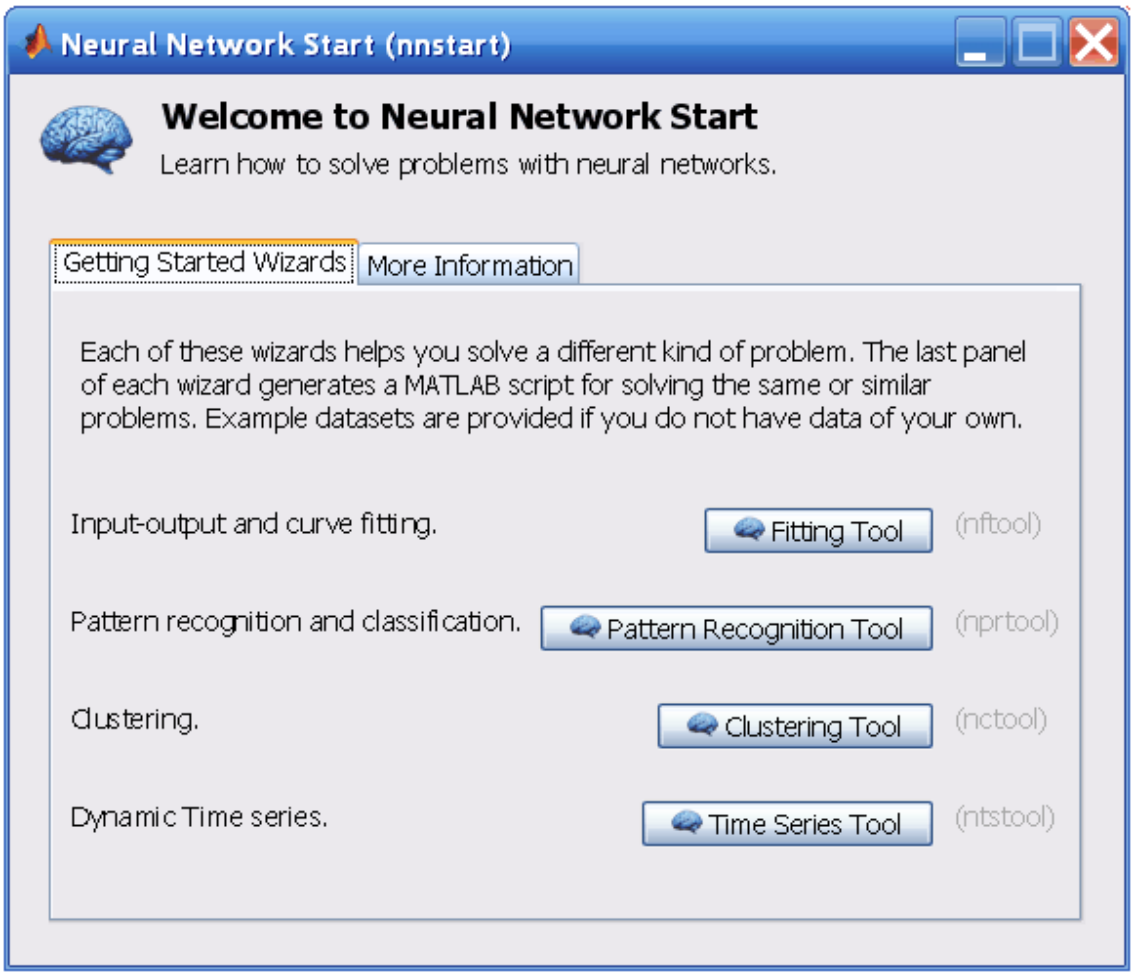


Figura 34 Inicio de la herramienta para redes neuronales

En este menú se pueden ver 4 opciones a elegir, cada una crea una red neuronal distinta y con fines distintos:

- Fitting tool: esta herramienta generara una red neuronal que relacionara variables matemáticas con sus resultados. Es muy útil para estimaciones de variables financieras como subida y bajada de acciones, precios de

inmuebles. Generalmente se usan para conceptos matemáticos relacionados con las finanzas.

- Pattern Recoginition Tool: la herramienta nos permite crear redes que sean capaces de distinguir patrones dentro de imágenes, para ello se utilizarán las características de los objetos e imágenes ya etiquetadas.

- Clustering Tool: esta red nos da grupos de objetos según sus características, se utilizan a la hora de encontrar segmentaciones de mercado, minería de datos y análisis bioinformáticos basados en algoritmos genéticos.

- Time Series Tool: con esta opción se genera una red que nos permita estimar datos de una serie temporal, introduciendo los datos históricos. Los análisis financieros se suelen basar en líneas temporales, con lo que esta herramienta es de gran ayuda en las futuras inversiones.

Como se puede observar la herramienta de redes neuronales es muy versátil y permite generar muchos tipos de funciones. En este proyecto se utilizara la herramienta Pattern Recognition Tool, ya que el objetivo es conseguir la identificación de patrones en la imagen. Con lo que seleccionaremos esa opción, dando lugar a la Fig. 35.

La Fig. 35 muestra un menú que ofrece información sobre la estructura que sigue la red, se observa que se creara una red de dos capas, la primera será la capa con las neuronas ocultas y una capa de salida. También ofrece ejemplos sobre el uso de este tipo de redes, en cierta manera esta aplicación [20] se puede utilizar como tutorial, en las demás opciones también se ofrece esta información para el usuario.

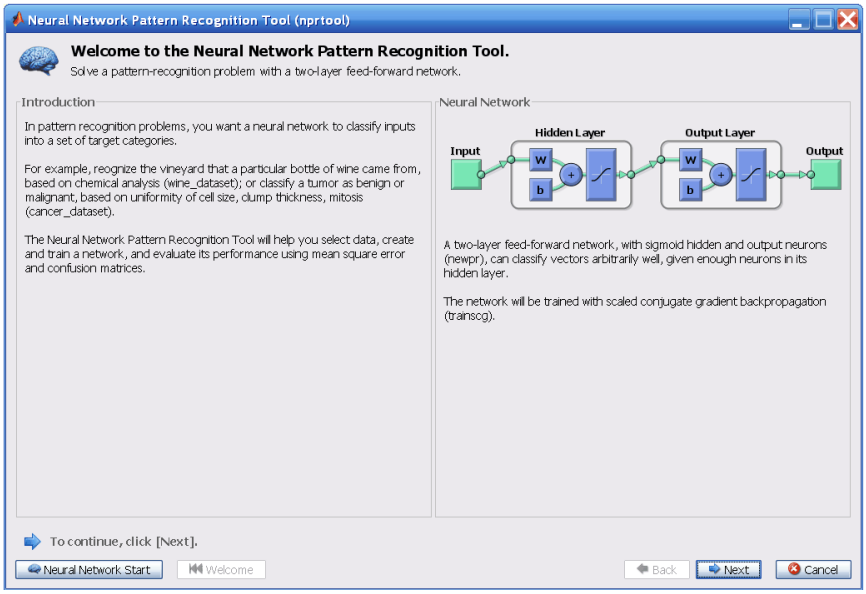


Figura 35 Herramienta Pattern Recognition Tool

Pulsando en el botón next accederemos al menú de selección de datos, Fig. 36, aquí se seleccionarán los datos a introducir, los inputs y los targets que hemos obtenido en la preparación de datos. Se podrán introducir las muestras por filas o por columnas, en este caso cada muestra estará representada por una columna. Y se utilizan matrices de 192.778 muestras. Con un mayor número de muestras y entornos se obtendrá una mejor red neuronal. Esta ventana también da la opción de generar una red neuronal a partir de datos ya creados por Matlab a modo de ejemplo.

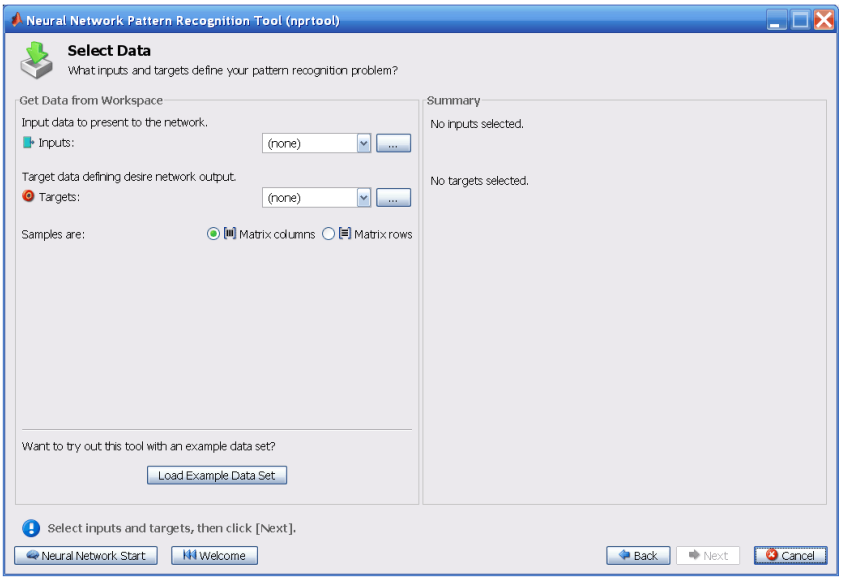


Figura 36 Introducción de datos para la red neuronal

Como se explica en la sección 3.5.1 las redes neuronales tiene tres fases entrenamiento, validación y test. El porcentaje de datos que se usara para cada una de las fases se seleccionará en la siguiente ventana, Fig. 37. Hay que tener en cuenta que los datos deben estar equitativamente repartidos en las muestras, esto quiere decir que es aconsejable que haya aproximadamente los mismo datos para todas las categorías de etiquetas, en nuestro caso se ha intentado realizar de esta manera. Pero debido a la gran cantidad de datos de oclusión y calzada hay muchos más datos sobre estas etiquetas, que de las demás etiquetas. Matlab seleccionara los datos para cada fase de una manera aleatoria, de manera que cada vez que se genere una red neuronal esta será diferente para los mismos datos, y sus relaciones entre neuronas también. Por eso es necesario crear varias redes neuronales y seleccionar la que de mejores resultados.

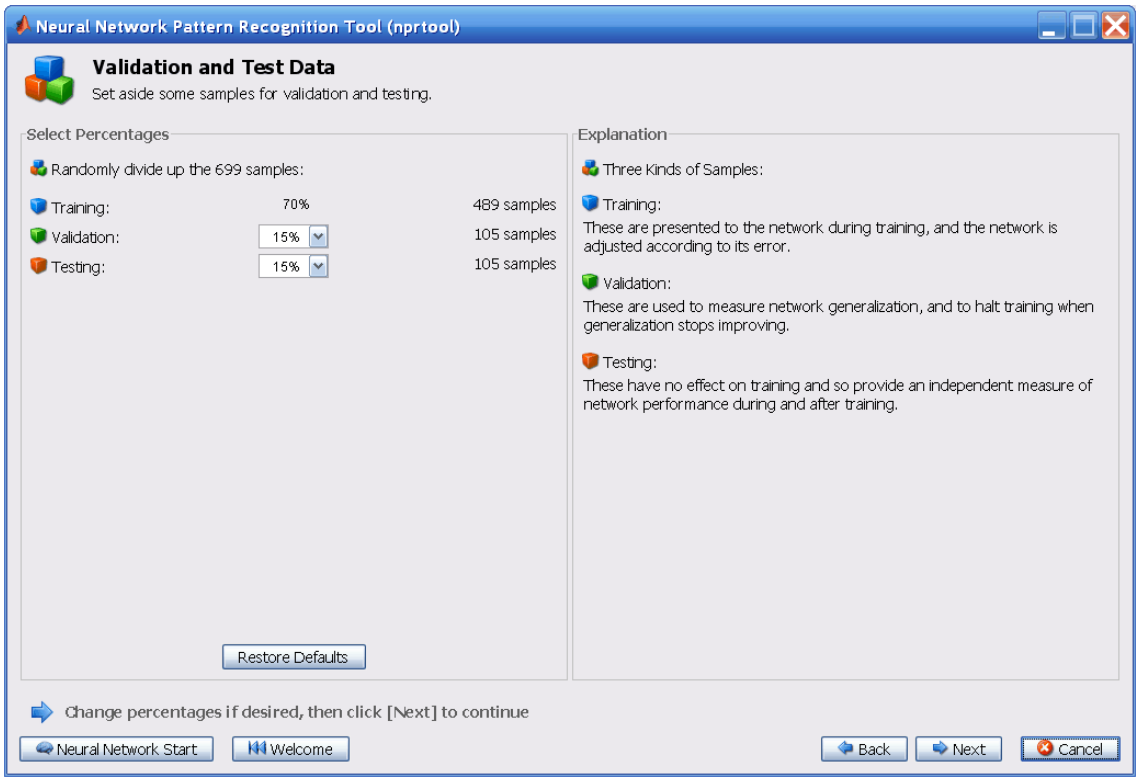


Figura 37 Selección del porcentaje para cada fase.

La siguiente ventana, Fig. 38, nos ofrece la opción de elegir el número de neuronas de la capa oculta, esta elección parece trivial, ya que simplemente se trata de introducir un número, es muy importante porque dependiendo del número de neuronas, se pueden obtener grandes diferencias en los resultados. Si se elige un

número bajo de neuronas, las relaciones que se crearan entre ellas no serán los suficientemente críticas, para que la red sea capaz de distinguir los patrones de cada objeto. Y si se introducen un número demasiado alto, la red generará demasiadas relaciones y el porcentaje de acierto bajara debido al *over-fitting*. El *over-fitting* se da cuando tienes un exceso de información y la red comienza a realizar relaciones contradictorias, esto hace que se pierda precisión.

Este número está altamente relacionado con el número de características que se introducen por muestra. Cuantas más características tengan, más neuronas ocultas se pueden introducir, ya que se pueden crear más relaciones entre las características al haber un número más alto de estas.

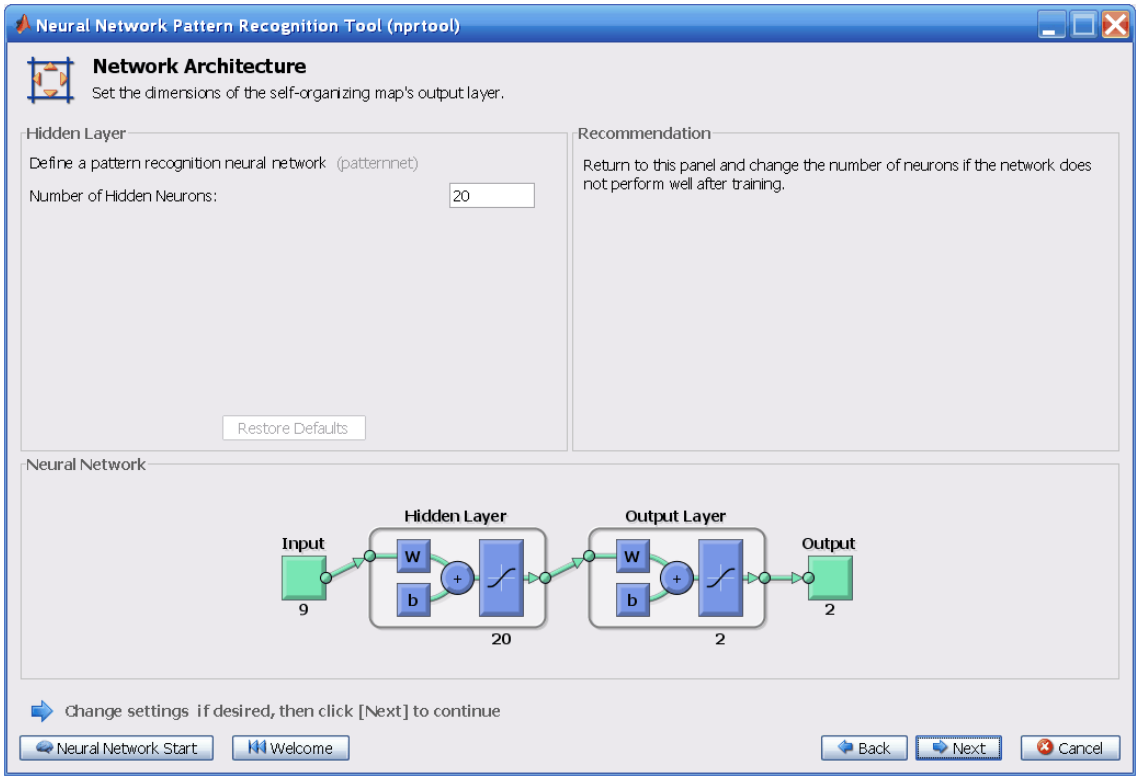


Figura 38 Introducción del número de neuronas

Una vez elegido el número de neuronas, se realizará el entrenamiento de la red, la Fig. 39 nos permitirá iniciar el entrenamiento, y una vez acabado este también nos mostrara las matrices de confusión para esa red. En caso de un mal resultado también se podrá reiniciar el entrenamiento.

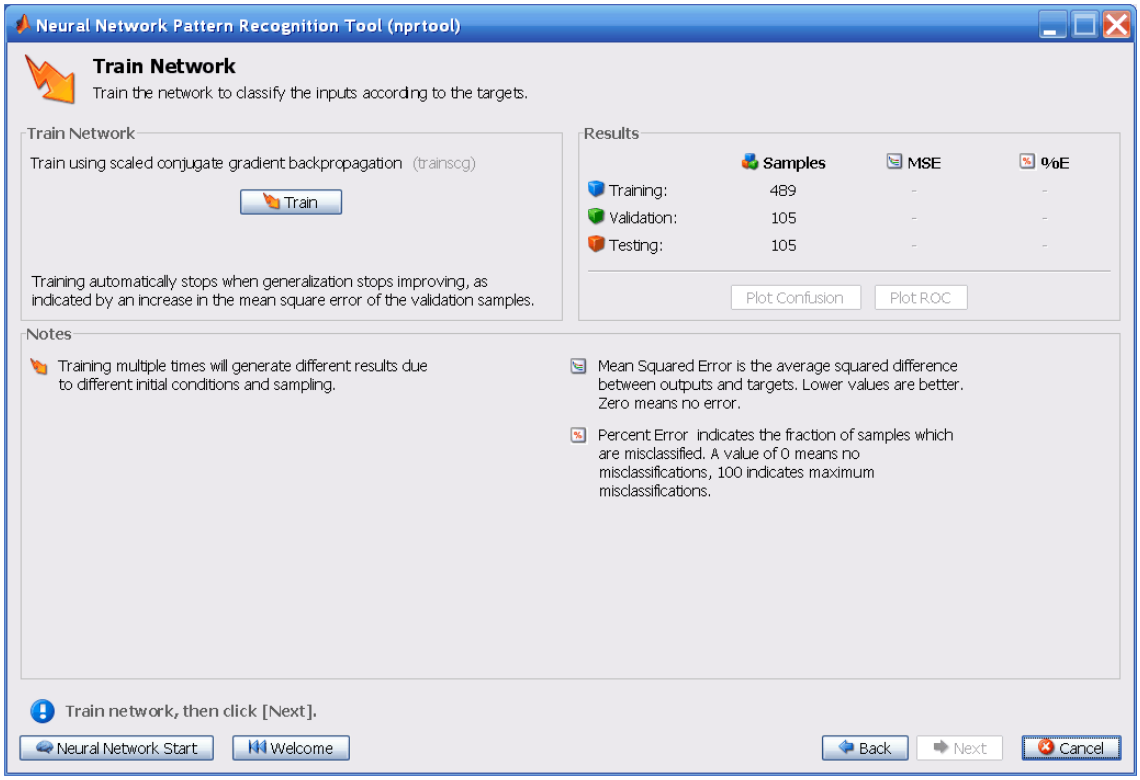


Figura 39 Ventana de entrenamiento

Una vez iniciado el entrenamiento aparece una ventana de progreso, Fig. 40, en las que aparece la estructura de la red neuronal. Y las funciones de división de datos, entrenamiento, rendimiento y derivada. También nos ofrece información en tiempo real del número de iteraciones hasta un máximo de 1000, el tiempo transcurrido desde el inicio del entrenamiento, el rendimiento de la red, el gradiente de la función derivada, y las validaciones, se realizaran 6 validaciones. También nos da la opción de ver los datos en tiempo real de la matriz de confusión, rendimiento, estado del entrenamiento, histograma de errores y regresión. Con la barra inferior se puede seleccionar el intervalo de iteraciones para que se refresquen los datos.

El cálculo de la red neuronal lleva un tiempo variable y es posible que el entrenamiento de la red de un resultado poco satisfactorio. Si se realiza un seguimiento de la matriz de confusión en tiempo real se puede detener el entrenamiento antes de que este acabe, esto es útil si se observa que los datos están muy sesgados. Esto ahorrará mucho tiempo de cálculo ya que las redes que no se estén entrenando bien se cancelaran.

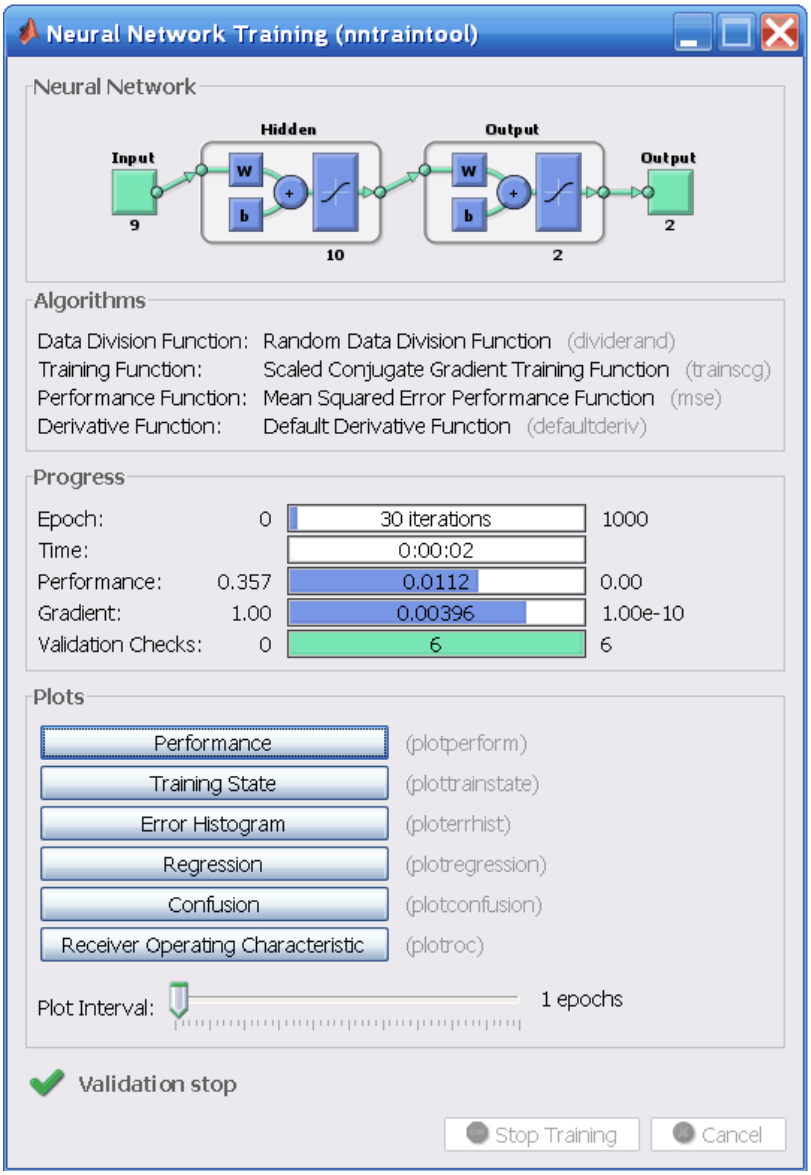


Figura 40 Ventana de entrenamiento

El entrenamiento finaliza cuando se han completado las 1000 iteraciones, que como máximo puede realizar, o cuando las 6 validaciones que tiene que cumplir la red son correctas. Muchos de los entrenamientos no son satisfactorios, ya que es posible que una gran cantidad de datos de una categoría hayan sido elegidos aleatoriamente en la parte de validación o test. Esto hace que aparezcan categorías sin datos de entrenamiento con lo cual la red los ignora y el entrenamiento acaba rápidamente. Esta situación genera una red neuronal que no nos sirve y hay que volver a realizar el entrenamiento. Cuando el entrenamiento ha terminado se pueden ver los resultados en la matriz de confusión. Como ejemplo podemos ver la Fig. 41 muestra una matriz de confusión para el ejemplo de red neuronal de Matlab.

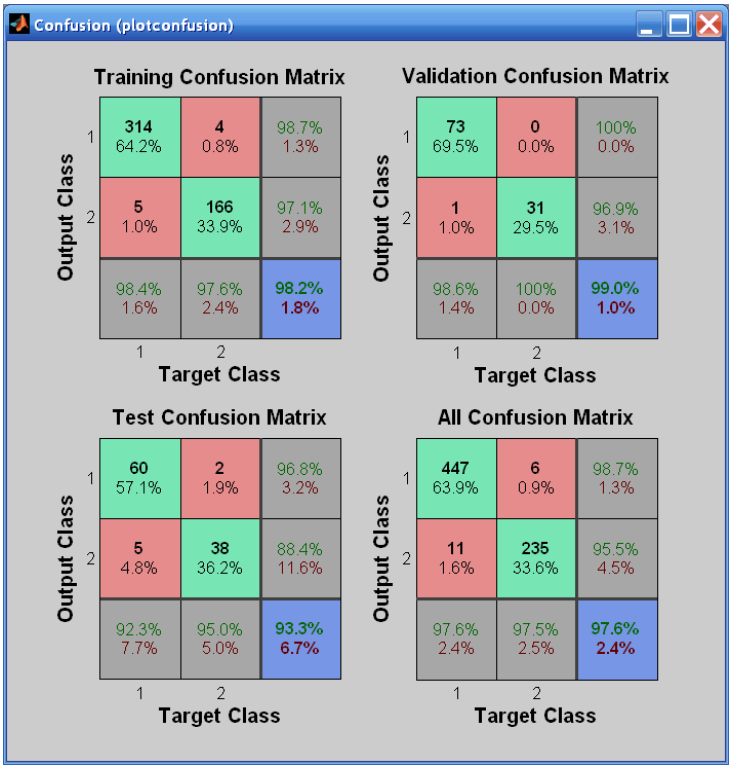


Figura 41 Matrices de confusión

La Fig. 42 muestra un menú, que nos permite seleccionar una serie de operaciones sobre la red antes de guardarla. La primera opción que nos da este menú es la opción de reentrenar la red desde cero, esto significa que la red elegirá otros datos para el entrenamiento, validación y tests. La siguiente opción es redimensionar la red neuronal esto significa que se elegirán de nuevo el número de neuronas que se usaran en la red y se realizara el entrenamiento otra vez. También da la opción de realizar el nuevo entrenamiento con un conjunto de datos más amplio o distinto. Y por último se pueden generar más tests para la red introduciendo más datos.

Por ultimo deberemos guardar los resultados en el entorno de Matlab, como se puede ver en la Fig. 43. El programa también da la opción de genera el script de la red neuronal. Esto permitirá reproducir esta red en otro ordenador. La red que se genera es igual ya que se deberán usar los mismos datos y este script seleccionara las misma muestras para el entrenamiento, validación y test. Se puede guardar también los outputs, los errores y la información sobre la red.

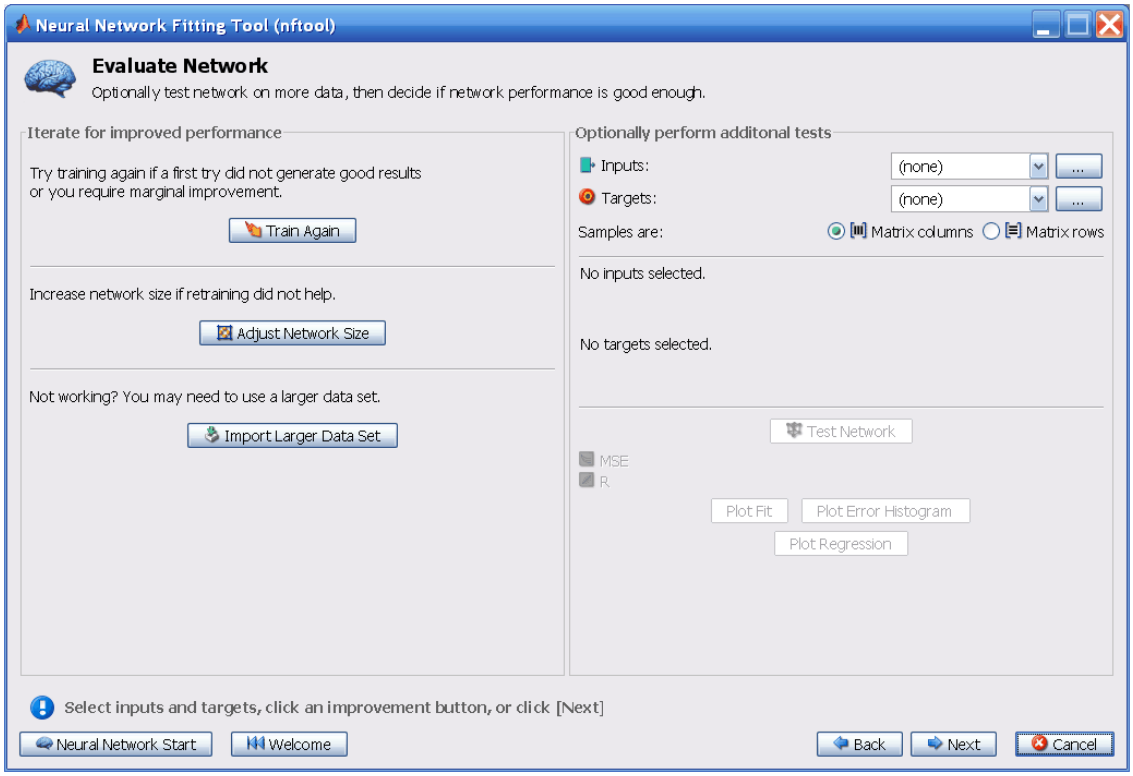


Figura 42 Menú después del entrenamiento

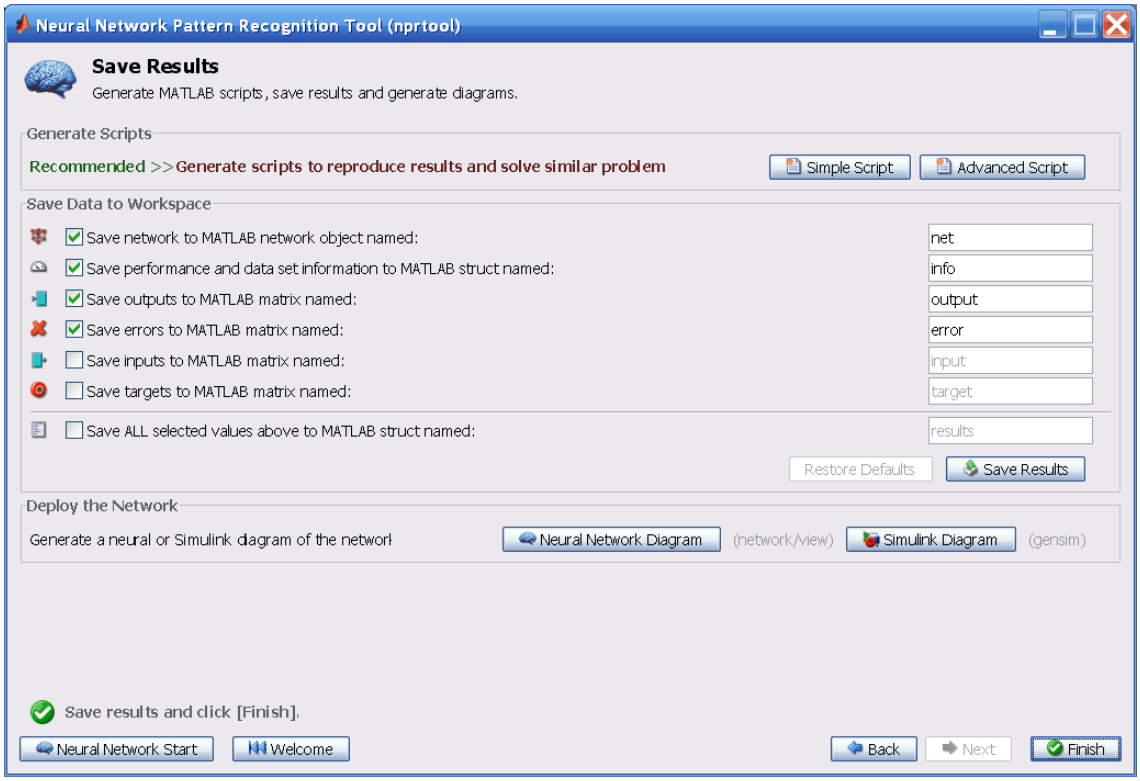


Figura 43 Guardado de red

4.3.Tratamiento de resultados

Primero se obtendrá el u_disparity de la imagen a etiquetar. Esto es necesario porque los datos que se obtienen de la red se usan para etiquetar el u_disparity de la imagen. Por eso se deberá obtener estos valores para etiquetar sobre ellos.

Cuando se obtiene los datos de input para la red del u_disparity se usará la función de Matlab sim, los datos tienen la misma forma de que los inputs de la red neuronal, la cual tiene la siguiente estructura:

[Y,Pf,Af,E,perf] = sim(net,P,Pi,Ai,T)

Los datos que se introducirán son net que hace referencia a la red que se empleará para el etiquetado, y P que son los datos de u_disparity a introducir. Los datos que obtendremos son el parámetro Y. Esto nos devolverá un tanto por uno con la probabilidad de que ese valor pertenezca a esa categoría. Como se puede observar en la Fig. 44, puede haber un porcentaje para más de una categoría. Cada columna representa un pixel y su etiqueta, es decir para la columna 19826 tiene la posibilidad de ser oclusión en un 96,99% y de ser calzada en un 4,21%.

```
Columns 19826 through 19838

  0.9699    0.9517    0.9771    0.9779    0.9695
  0.0421    0.0476    0.0386    0.0384    0.0430
  0.0000    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.0000
```

Figura 44 Extracto de los resultados de la red neuronal

Como tenemos que tener una sola categoría por pixel, el criterio que se sigue es el de seleccionar el valor de probabilidad más alto. De esta manera obtendremos los datos ya asignados a una categoría, Fig. 45.

```
Columns 19825 through 19840

     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

Figura 45 Extracto de los datos con matriz asignados.

Se realizara el tratamiento de los datos obtenidos de la red. Los datos que ofrece esta es una matriz en la que cada columna es un pixel del u_disparity, con lo que esta matriz tendrá un tamaño de 6 filas por 14880 columnas (31x480). El orden en el que están los datos es, partiendo de la esquina superior izquierda se recorrerá primero de izquierda a derecha y cuando se termina la fila se pasara a la siguiente.

Una vez obtenido los resultados de la red, estos se reagruparan otra vez para formar un u_disparity etiquetado. Se utiliza el sistema RGB debido a su simplicidad. El proceso comienza creando una matriz de 3 dimensiones (31x480x3), ya que las imágenes en RGB necesitan tres matrices de pixeles para definir los colores, una para el rojo (R), otra para el verde (G) y por último la matriz del azul (B). A cada matriz se le otorgara un valor para componer los colores de la imagen, los colores elegidos son:

- Oclusión: Blanco, color RGB [255,255,255]
- Calzada: Verde, color RGB [0,255,0]
- Vehículo: Azul claro, color RGB [0,255,242]
- Peatones: Amarillo, color RGB [255,233,0]
- Cielo: Azul oscuro, color RGB [0,0,255]
- Obstáculo: Rojo, color RGB [255,0,0]

La codificación en Matlab del RGB es de 0 a 1 con lo que hay que realizar el ajuste de cada color para que de un buen resultado. Finalmente el resultado del etiquetado del u_disparity es el mostrado en la Fig. 46

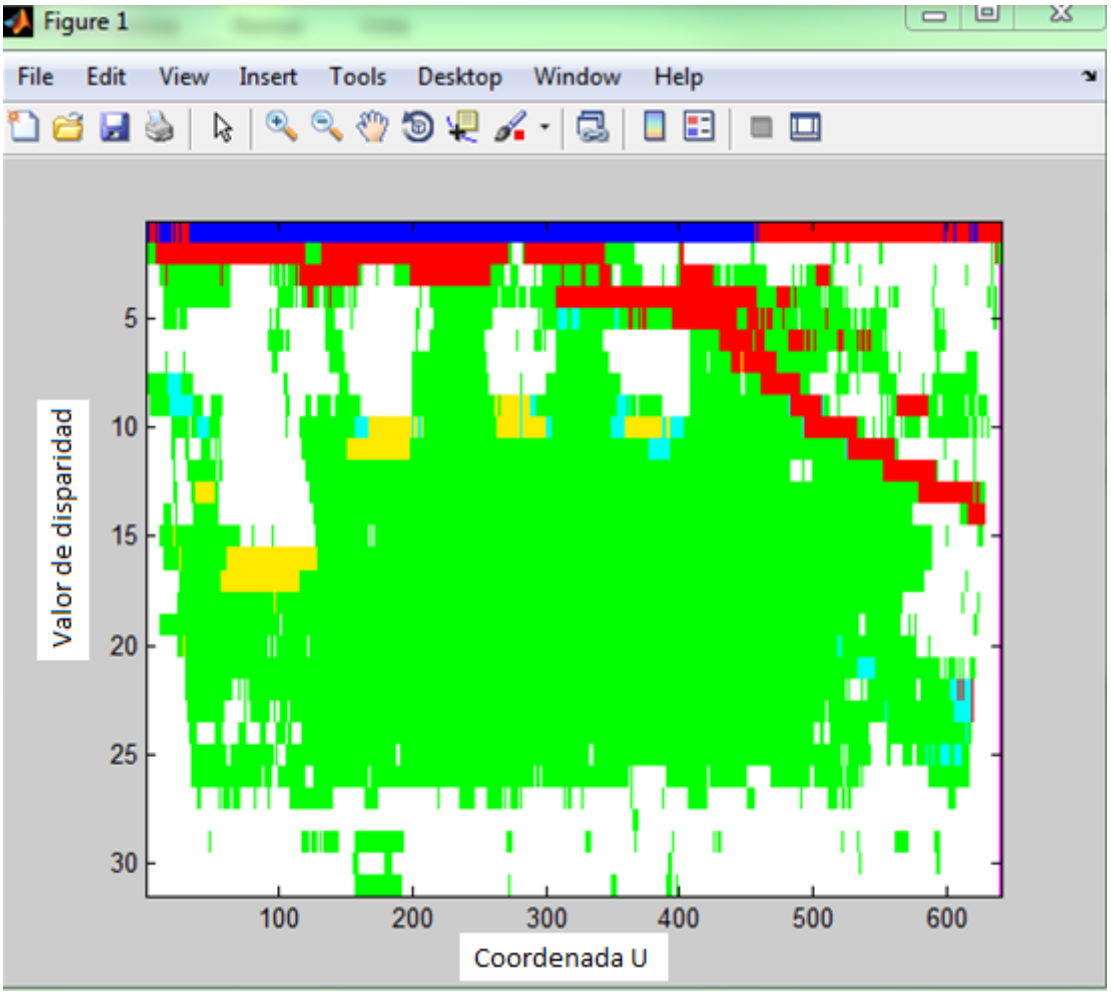


Figura 46 U_disparity coloreado.

Como se puede observar la gran mayoría de los pixeles corresponden con la calzada, los valores de cielo siempre están colocados en el menor valor de disparidad. En la Fig. 46 el eje horizontal corresponde con la coordenada U del u_disparity, que esta a su vez coincide con la posición U del objeto en la imagen. Y el eje vertical corresponde con los valores de disparidad.

El siguiente paso es asociar a cada punto (pixel) del u_disparity con los correspondientes en el mapa de disparidad. Esta tarea aunque parece sencilla no lo es, ya que no tenemos la posición en Y de los puntos de disparidad. En el u_disparity solo se tiene la información de la coordenada U, el valor de disparidad (d) y la concentración de puntos en esa disparidad. Como se tiene los datos del mapa de disparidad y del u_disparity etiquetado. Simplemente se iniciará una matriz a 0 de 3 dimensiones del alto y ancho de la imagen, es decir 640x480x3 para tener las imágenes

en color. Después se volverá a calcular el u_disparity, pero cuando se selecciona el pixel del mapa de disparidad para ser añadido al u_disparity, esto nos da la coordenada del u_disparity. Una vez obtenido este valor, es muy sencillo copiar los valores del u_disparity coloreado y copiarlo en la matriz de 3 dimensiones. Con esto se obtendrá el mapa de disparidad coloreado con las etiquetas correspondientes.

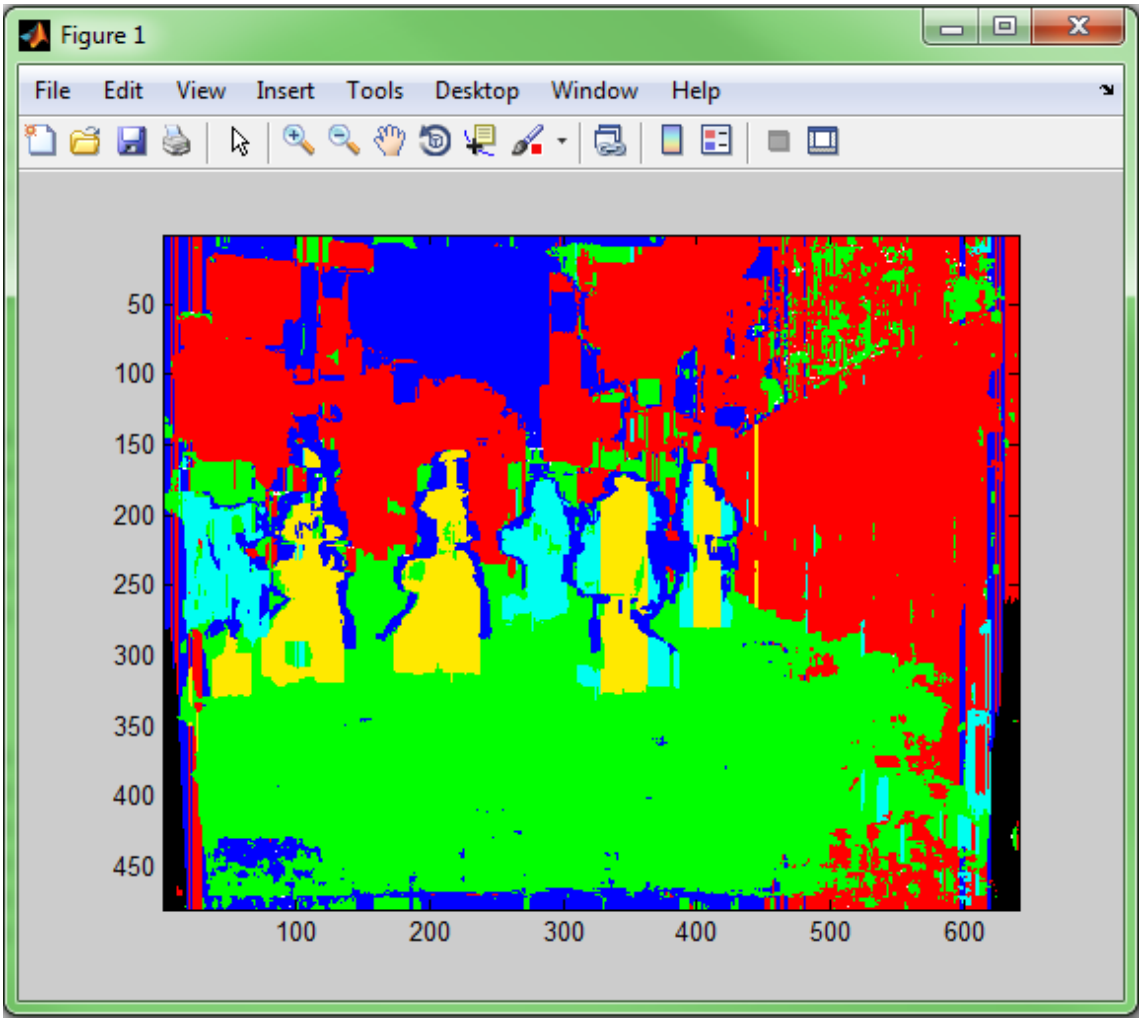


Figura 47 Mapa de disparidad etiquetado

Como se puede observar, Fig. 47, ya se pueden intuir como es la imagen. El último paso es trasladar los datos del mapa de disparidad a la imagen real. Debido a que el mapa de disparidad está formado por dos imágenes, al colorear estas habrá cierto desfase. Pero nos dará una información más fácil de interpretar debido a que se etiqueta sobre la imagen real.

En esta parte se realiza cogiendo la imagen derecha o izquierda, se creara una matriz de 3 dimensiones en la que estará copiada la imagen 3 veces. Como la imagen es de 640x480 pixeles, está estará en cada nivel de la matriz. Al tener valores de color en blanco y negro esto significa que los valores de RGB son parecidos. Para no perder la definición de la imagen esta no se coloreara por completo sino que simplemente se teñirán ligeramente los objetos del color de su etiqueta. Esto se realiza incrementado el valor de cada capa según el color que se tenga. Si se quiere que la imagen tenga un tono rojo se tendrá que incrementar valor del pixel rojo, esto corresponde con las coordenadas i,j del pixel y su primera capa (i,j,1). Y así sucesivamente, para colores distintitos del rojo, verde o azul habrá que cambiar los valores en distintos niveles. El resultado final se puede apreciar en la Fig. 48.

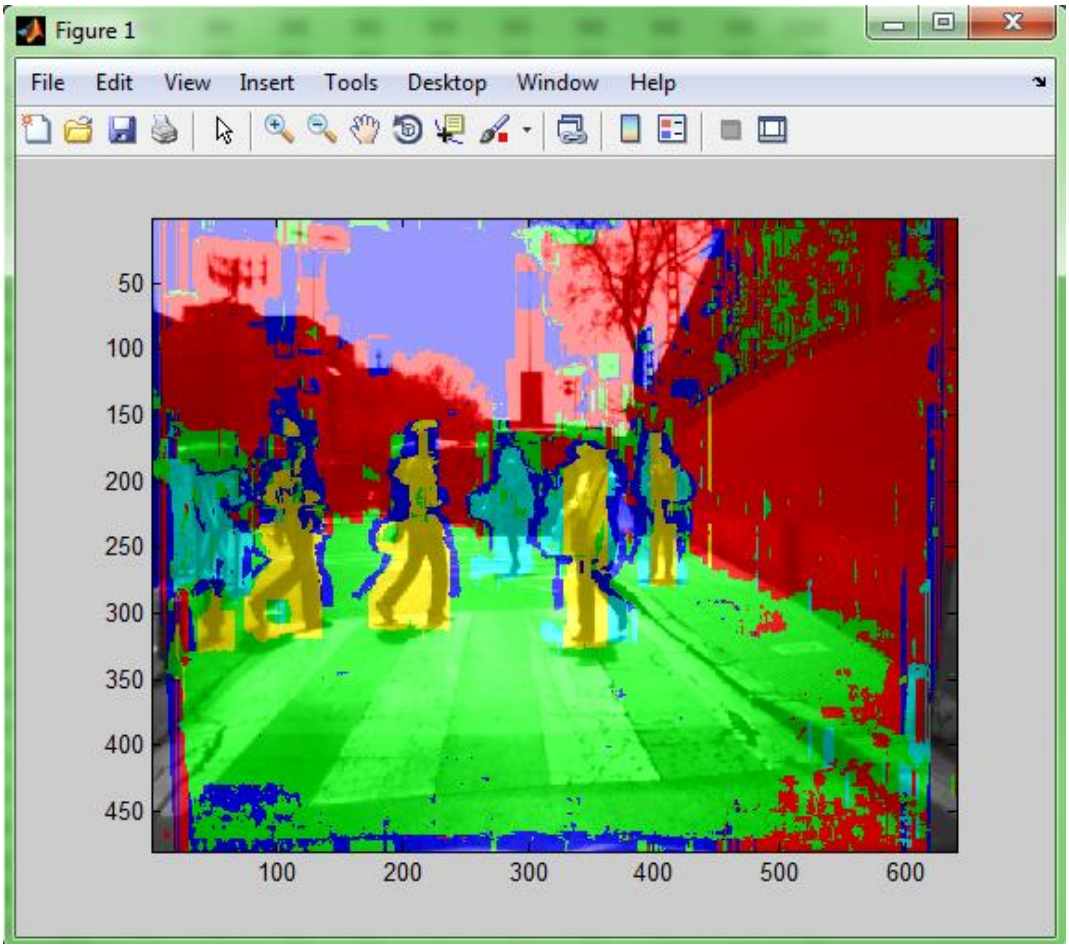


Figura 48 Resultado final de la imagen

5. Resultados

Para este proyecto se han entrenado distintas redes neuronales, con las mismas muestras pero variando entre LBP compacto y extendido. También variando el número de neuronas por red, esto arrojara distintos resultados, tanto en nivel de acierto como en tipo de clasificación de los objetos.

Los datos que utilizaremos serán 192778 pixeles extraídos de distintos escenarios. En el etiquetado de imágenes completas se utilizan 13 imágenes con distintos escenarios. De estas imágenes se eliminaran ciertas muestras al azar de datos de calzada y oclusión. Esto se realizar para compensar el exceso de información correspondiente a estos datos.

También se usaran datos de dos secuencias de video, la primera secuencia corresponde con imágenes con una alto contenido de peatones, estas imágenes se etiquetaran por partes para añadir el menor número de datos posible de calzada y oclusión. De estos datos solo se añadirán los pixeles en contacto con los objeto. Y el segundo juego de imágenes que se usa, es una secuencia en la que se ven circular coches en distintas posiciones, tanto a la derecha como la izquierda y a distintas distancias.

Todos los datos han sido recogidos con una cámara estéreo montada sobre el salpicadero del coche.

Para la comprobación de las redes neuronales se usaran 4 imágenes en entornos que la red jamás ha procesado. Esta son las siguientes, Fig. 49.

(A)

(B)

(C)

(D)

Figura 49 Parte derecha de las imágenes de prueba

5.1.Primera red neuronal.

Para la primera red neuronal se selecciona un red neuronal de 10 neuronas ocultas. Se eligió 10 neuronas debido a que estamos introduciendo como datos de partida el conjunto de datos de posición en X, nivel de disparidad, concentración de puntos (valor de intensidad del u_disparity) y por último se usara el LBP compacto. Con lo que estamos introduciendo solo 4 características. La relación 4 características – 10 neuronas ocultas, es lo suficientemente alto para crear las relaciones necesarias entre las características y tampoco perder información por over-fitting. La Fig. 50 muestra cuando se finaliza el entrenamiento de esta red.

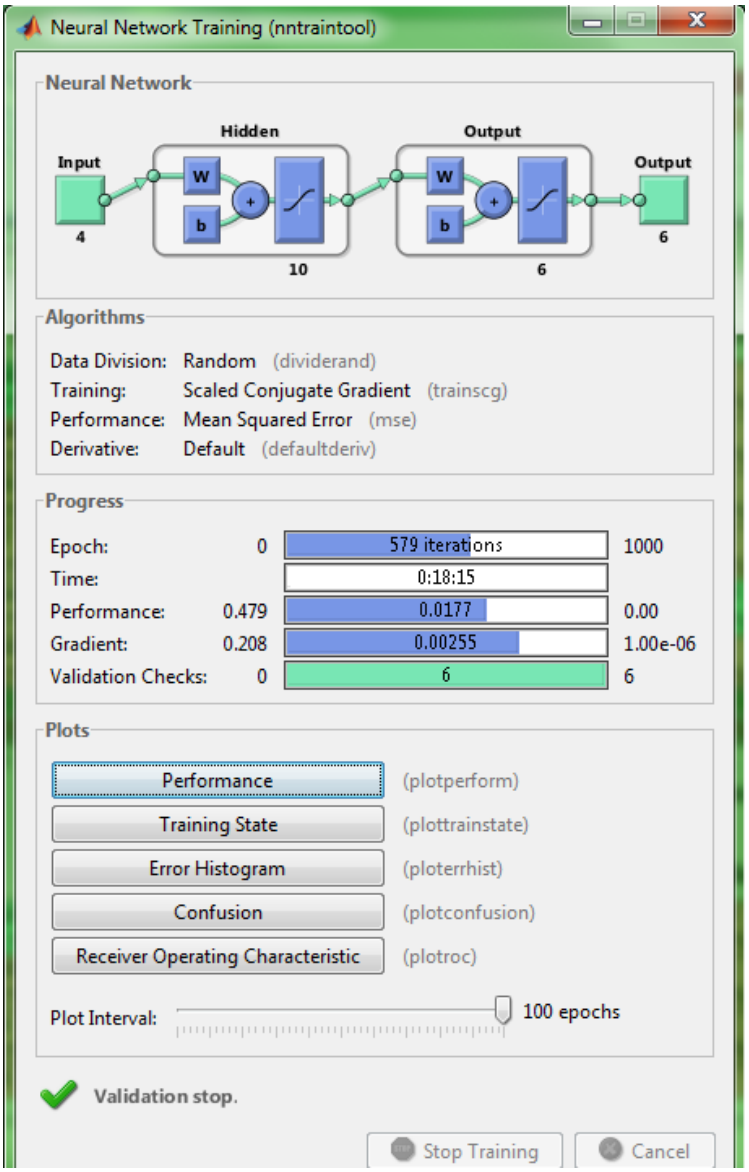


Figura 50 Entrenamiento red 10 neuronas

Como se puede observar en la ventana de entrenamiento este termina cuando a realizado 579 iteraciones. Acaba antes de llegar a las 1000 debido a que las 6 validaciones se han realizado con éxito. Se puede observar que el tiempo que ha tardado el sistema en realizar esta operación ha sido 18 minutos y 15 segundos. Este tiempo es bastante bajo debido al bajo número de neuronas que se utilizan, cuantas más neuronas haya más se tardará.

Los resultados para la matriz de confusión son los siguientes, Fig. 51.

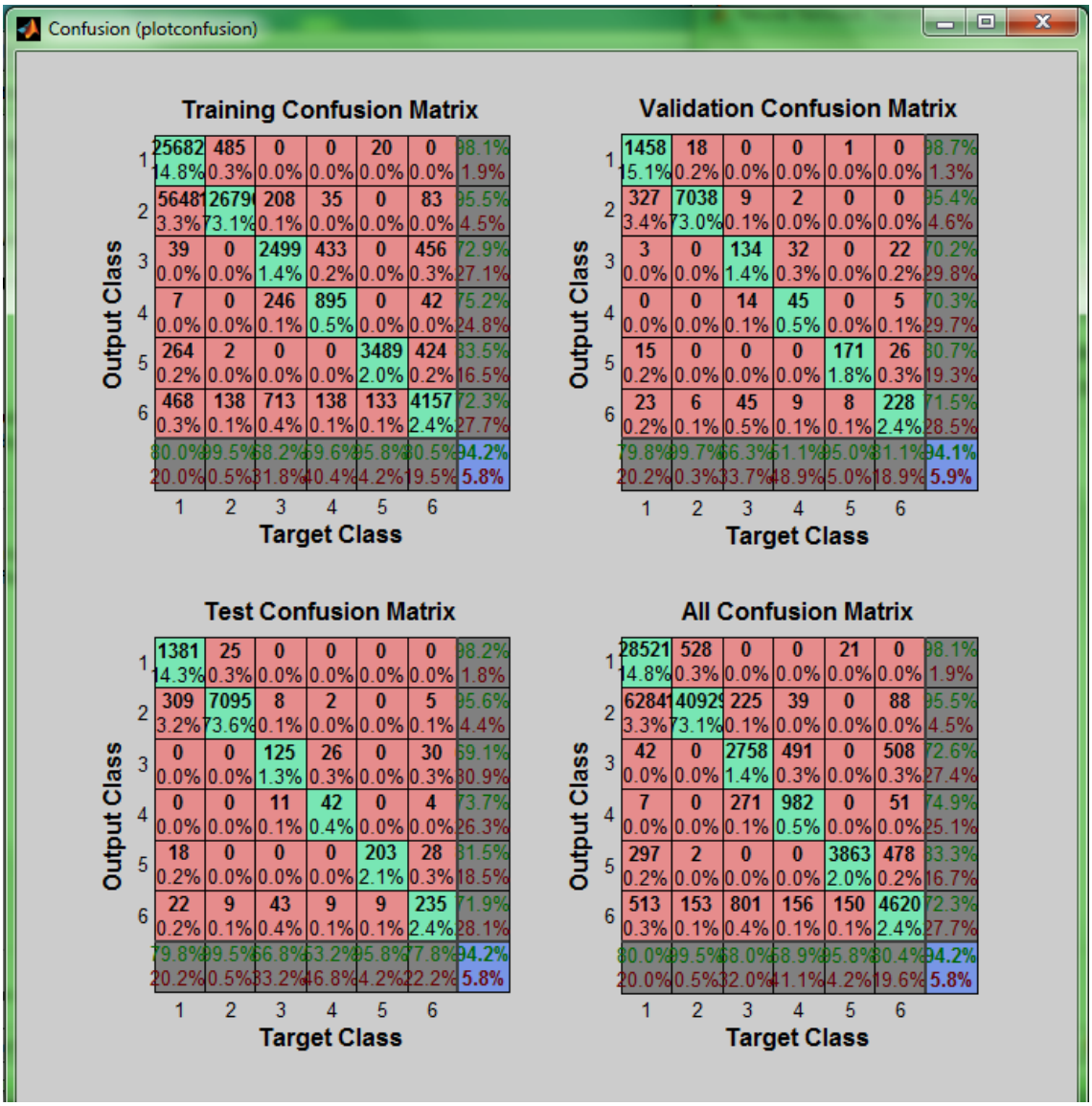


Figura 51 Matriz de confusión 10 neuronas

Las filas nos indican el nivel de acierto para esa categoría, es decir cuantos de los puntos de esa categoría no han sido clasificados. Y las columnas nos ofrecen cuantos puntos de otras categorías han sido etiquetados como puntos de esta. Para las 4 matrices es igual, la matriz de entrenamiento nos ofrece el nivel de cierto que se consigue en la fase de entrenamiento, la de validación en la fase de validación y por último la matriz de test que nos ofrece los resultados finales. La cuarta matriz nos sirve para observar el nivel de acierto en las tres fases en conjunto.

Como se puede observar en la matriz los resultados para cada categoría, la matriz más importante es la matriz de confusión de test. Esta matriz nos dice el porcentaje de acierto en las distintas categorías:

- 1 = Oclusión: 98,2% de acierto
- 2 = Calzada: 95,6% de acierto
- 3 = Vehículo: 69,1% de acierto
- 4 = Peatón: 73,7% de acierto
- 5 = Cielo: 81,5% de acierto
- 6 = Obstáculo: 71,9% de acierto

A la vista de estos porcentajes se ve que la red obtiene los mejores resultados para oclusión y calzada, pero para las demás categorías no son lo suficientemente buenos como para ser una red definitiva. Esta red se comprobara con 4 imágenes (Fig. 49) que no han sido utilizadas en ninguna de las fases implicadas en el entrenamiento de la red.

Se analizaran las imágenes por separado y después se realizara una comparación entre todas las redes.

Figura 52 Resultado imagen (A) 10 neuronas

En esta imagen, Fig. 52, se puede observar que el etiquetado de la calzada es bastante preciso. Los fallos que comete la red al confundir obstáculo con calzada son debidos al error de la matriz de confusión, ya que la red no es 100 % precisa. En cuanto al cielo se puede observar que se realiza un etiquetado correcto de la zona de cielo, pero aparecen puntos dispersos por toda la imagen. Esto es debido a que los valores de oclusión que se generar alrededor de los objetos, debido a la diferencia entre las imágenes. Esta oclusión tiene la misma disparidad que el cielo, con lo cual la red no es capaz de diferenciar entre esta oclusión y el cielo, ya que los valores de oclusión blancos quedan tapados por los objetos y no se observan en la imagen. Todos aquellos que si son observables se etiquetan como cielo debido a su valor coincidente con el cielo.

En cuanto a los obstáculos etiquetados en rojo, se puede observar que se etiquetan correctamente en la gran mayoría, pero confunde un coche con obstáculo. Esto se debe a que en el mapa de disparidad en la imagen el túnel, el muro y la calzada están muy juntos en el u_disparity, Fig. 53. Estos pixeles están muy acumulados de manera que los objetos que no resalten sobre los demás son difíciles de distinguir. Como el coche es más bajo que el túnel por el que está pasando hace que en el u_disparity se vea como un objeto solo muy difícil de separar.

Con respecto al coche que se encuentra frontalmente se puede observar que hay una mezcla entre coche y peatón, esto se debe a que los peatones que se encuentran lejos del vehículo y por tanto del sistema estéreo tienen un perfil muy parecido al de los coches. Esto es debido a que los coches son más bajos que las persona, así que en la imagen tiene ciertos problemas para diferenciarlos. También hay que tener en cuenta las tasas de acierto de esta imagen que no es muy alto.

Figura 53 U_disparity imagen (A)

Por último se ve que hay una clara equivocación en el muro, esta red lo etiqueta directamente como una persona, debido a que la gran mayoría de los peatones que se utilizan en el etiquetado son peatones que están cerca de la cámara. Este muro está muy cerca de la cámara, en el u_disparity se aprecia el muro como la línea blanca que va desde la izquierda a la derecha. Al ser un perfil en diagonal esto hace que se confunda con los peatones, porque los peatones tienen un perfil muy característico en forma de pequeña línea horizontal o ligeramente inclinada, pero lo más importante es que tiene un valor alto de intensidad.

Hay cierto valores que no se etiquetan, esto se debe a que esos pixeles no aparecen en la imagen, debido a que esos puntos no están en el mapa de disparidad. Ya que son puntos que en este caso solo pertenecen a la imagen derecha de la cámara.

Los resultados para la imagen (B) son Fig. 54.

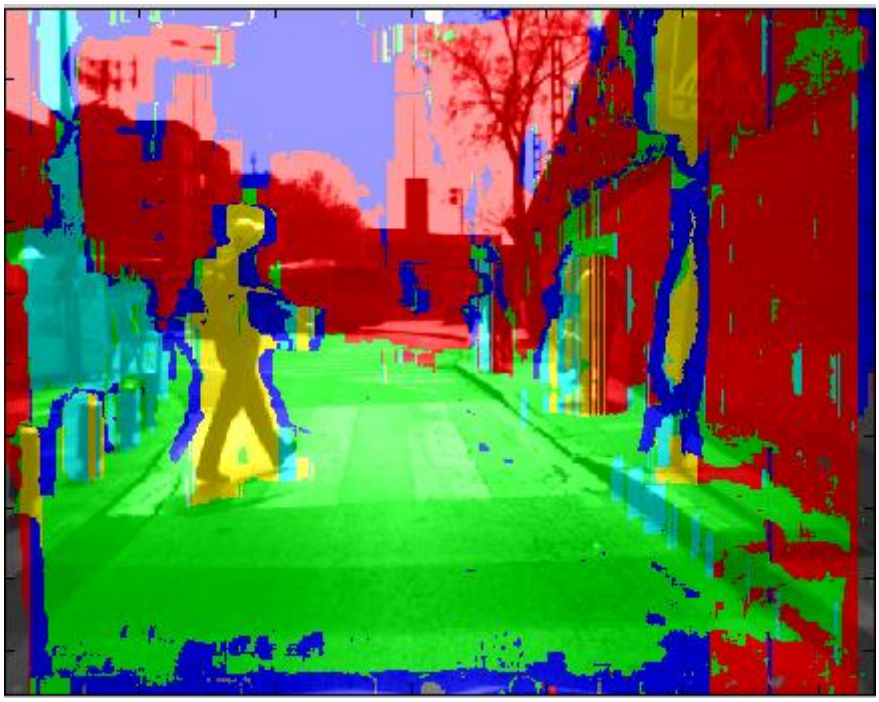


Figura 54 Resultado imagen (B) 10 neuronas

Para esta imagen se consiguen unos resultados más satisfactorios que para la anterior, se observa que se sigue teniendo el mismo problema con la oclusión y cielo. Pero en lo que respecta al resto de la imagen se observa un buen etiquetado de los obstáculos en general.

Respecto a los peatones se puede ver que el peatón que está cruzando la calle se etiqueta perfectamente. El peatón que se encuentra más a la derecha de la imagen se diferencia un poco, pero al estar muy lejos de la imagen hace que se confunda con el perfil de u_disparity, Fig. 55, de un coche. Además se encuentra muy cerca del muro lo que también hace que sea muy difícil diferenciarlo de este. Por ultimo el peatón que se encuentra más alejado apenas es etiquetado, y la etiqueta que se le asigna es la de coche debido a su alejamiento. También se produce un error en la señal de tráfico debido a que su perfil de disparidad es semejante al de un peatón.

Figura 55 U_disparity imagen (B)

También se produce un error de etiquetado con el cubo de basura, este es confundido con un coche debido a su forma, esta es similar a la de un coche esto se puede observar en el u_disparity, Fig. 55. Y por la misma razón se confunde los pivotes de la acera con coches o peatones. Todo es debido al u_disparity de la imagen y esos objetos.

Los resultados para la imagen (C), Fig. 56.

Figura 56 Resultados imagen (C) 10 neuronas

Esta imagen es similar a la anterior, se puede observar que lo errores son los mismos que para la imagen (B). Esto significa que el modo en que la red etiqueta la imagen es constante y sigue los mismos criterios. Se puede comprobar que la identificación de los peatones es bastante correcta, aunque sigue surgiendo el problema de la diferenciación entre peatones y vehículo, y ciertos obstáculos con los coches.

Figura 57 U_disparity imagen (C)

Los problemas que surgen con los peatones, es que en los extremos de los peatones son similares a las características de los coche en el u_disparity, Fig. 57.

Los resultados de la imagen (D) son, Fig. 58.

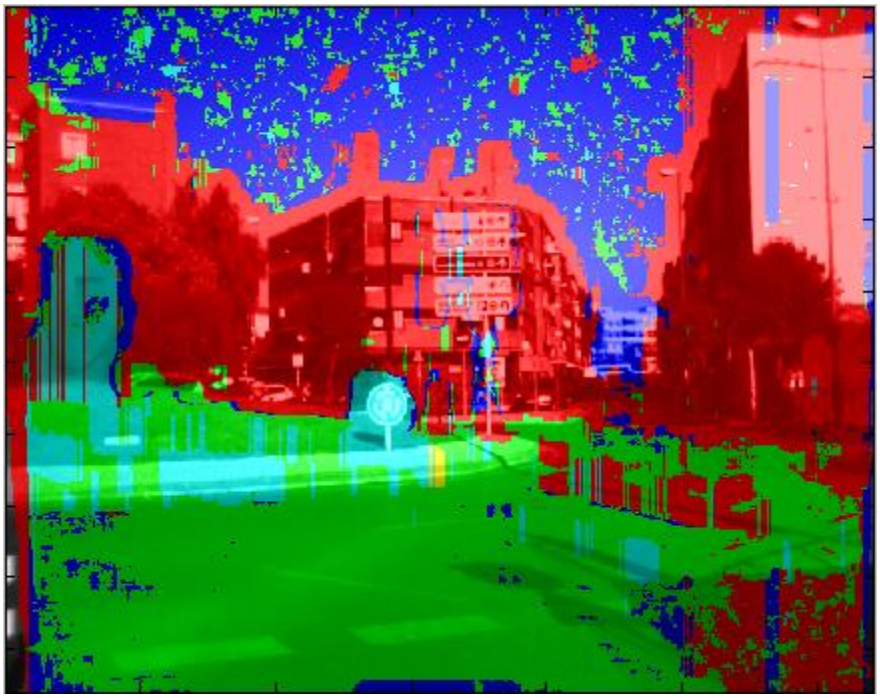


Figura 58 Resultados imagen (D) 10 neuronas

Esta imagen es mucho más simple que la anterior ya que solo contiene 3 categorías, ya que no hay ni peatones ni vehículos. Se puede observar que el error cometido es pequeño, todos estos errores se podrían atribuir a los porcentajes de acierto. La detección de la señal y la escultura como vehículo se debe a su perfil en el u_disparity, Fig. 59.

Figura 59 U_disparity imagen (D)

5.2. Segunda red neuronal.

Esta red usa las mismas muestra (LBP compacto) que la anterior pero se utilizan 30 neuronas ocultas, esto hace que se creen más relaciones entre las características mejorando el porcentaje de acierto. La finalización del entrenamiento fue la siguiente, Fig. 60.

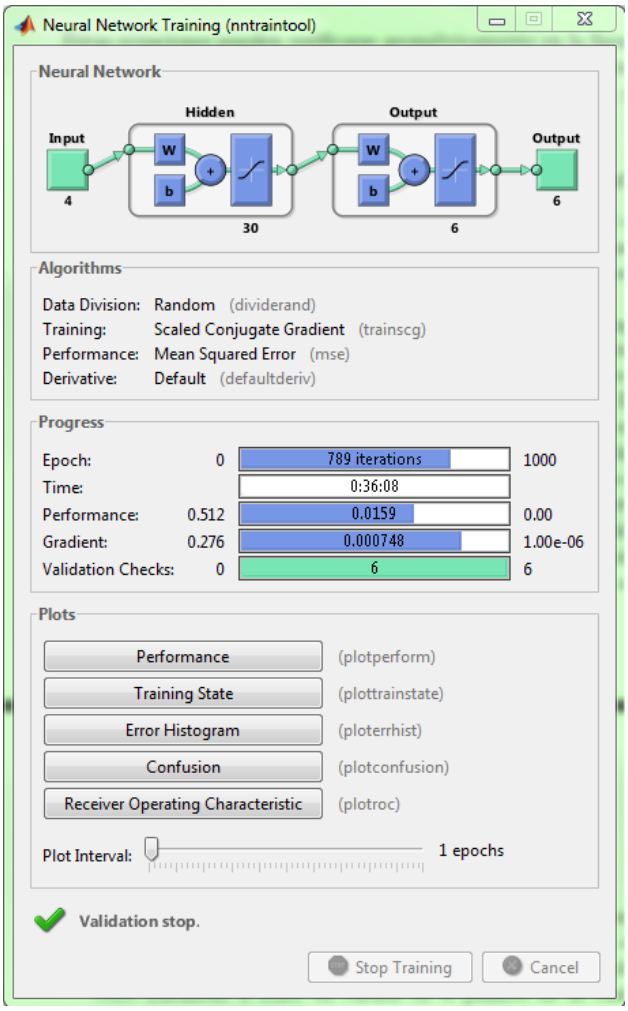


Figura 60 Entrenamiento red 30 neuronas

Se observa que el entrenamiento realiza 789 iteraciones hasta que se completan las 6 validaciones necesarias, para que se dé por bueno el entrenamiento. El tiempo empleado para este entrenamiento es el doble que el anterior, pasando de unos 18 minutos a 36 minutos. El ordenador tarda más en procesar está red debido a que contiene el triple de neuronas que la anterior.

Su matriz de confusión, Fig. 61, arroja unos resultados mejores que la anterior red neuronal.

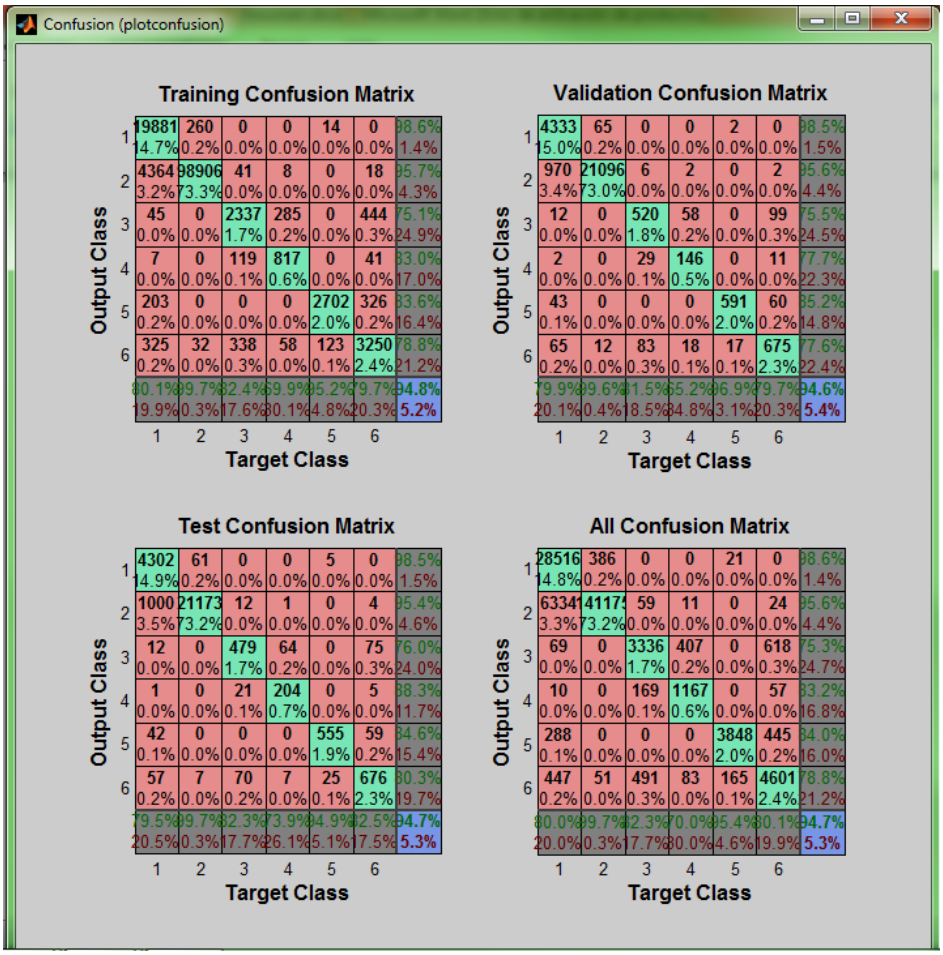


Figura 61 Matriz de confusión 30 neuronas

Los resultados para cada categoría son:

- 1 = Oclusión: 98,5% de acierto
- 2 = Calzada: 95,4% de acierto
- 3 = Vehículo: 76,0% de acierto
- 4 = Peatón: 88,3% de acierto
- 5 = Cielo: 84,6% de acierto
- 6 = Obstáculo: 80,3% de acierto

Los porcentajes de acierto han mejorado notablemente, esto se reflejará en las imágenes de prueba en una mejor identificación de los objetos.

Resultados para la imagen (A), Fig. 62.

Figura 62 Resultados imagen (A) 30 neuronas

Como se puede observar la identificación de la calzada mejora ligeramente, ya que se adentra más en la zona del túnel. El coche recibe un mejor etiquetado ya que este es más uniforme que el anterior. Respecto al muro se sigue teniendo la misma falsa identificación, que viene dada por la disposición de muro. Existe cierta confusión con los objetos como el edificio de la izquierda que antes era evaluado como obstáculo y ahora es evaluado como vehículo, esto se debe a las nuevas relaciones que establecen las neuronas ocultas.

Pero se puede asumir que el resultado ha mejorado ligeramente pero todavía existe fallos que se intentaran corregir en las siguientes redes.

Resultados de la imagen, Fig. 63.

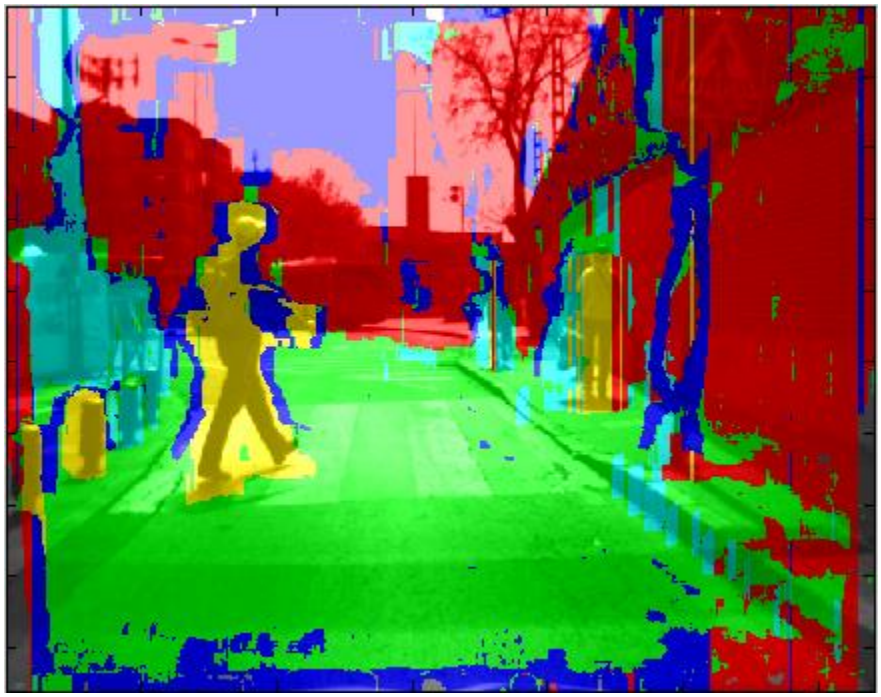


Figura 63 Resultados imagen (B) 30 neuronas

La Fig. 63 corrige ciertos fallos que aparecían en los resultados anteriores, se observa que el etiquetado del peatón de la izquierda es mejor y más definido, el peatón de la derecha es etiquetado casi por completo. Los fallos que comete se pueden atribuir perfectamente al porcentaje de acierto. Por último el peatón que antes casi no se detectaba es detectado aunque como vehículo, esto es debido a que se encuentra a una gran distancia.

Otra corrección importante es la de la señal de tráfico, que ahora está completamente etiquetada como obstáculo. Lo único que no se ha corregido ha sido el error del cubo de basura, pero esto es debido a que tienen una forma y disposición muy similar a la de un vehículo. Existen ciertos fallos menores como la detección de los pivotes como objetos no obstáculo.

En esta imagen sí que se aprecia una buena mejoría de los resultados respecto a la anterior prueba.

Resultados imagen (C), Fig. 64.

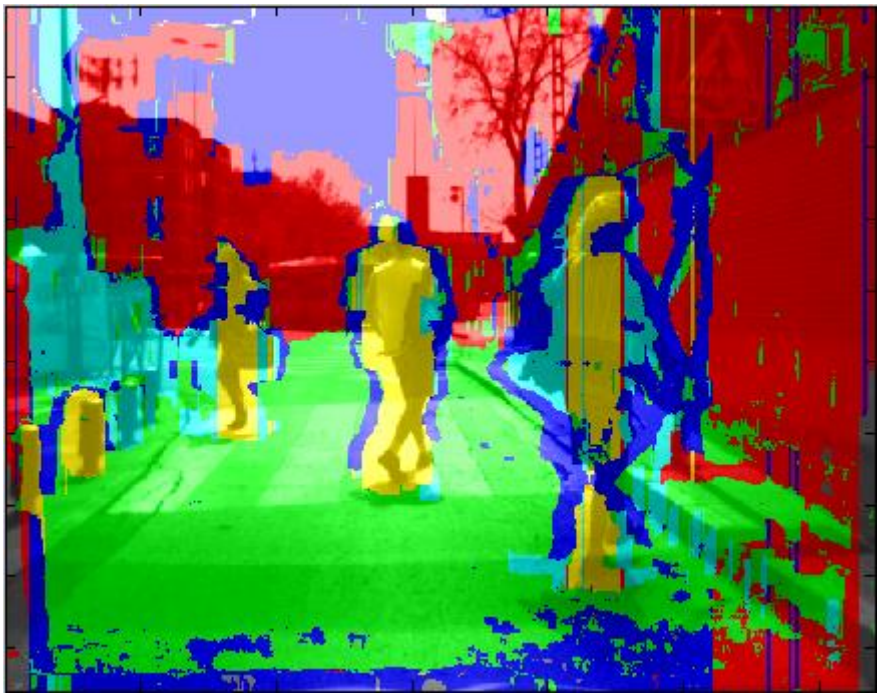


Figura 64 Resultados imagen (C) 30 neuronas

Como en la imagen anterior se aprecia las mismas mejoras respecto al etiquetado de los peatones, y de la detección de la señal, pero persiste el mal etiquetado del cubo de basura.

Resultados imagen (D), Fig. 65.

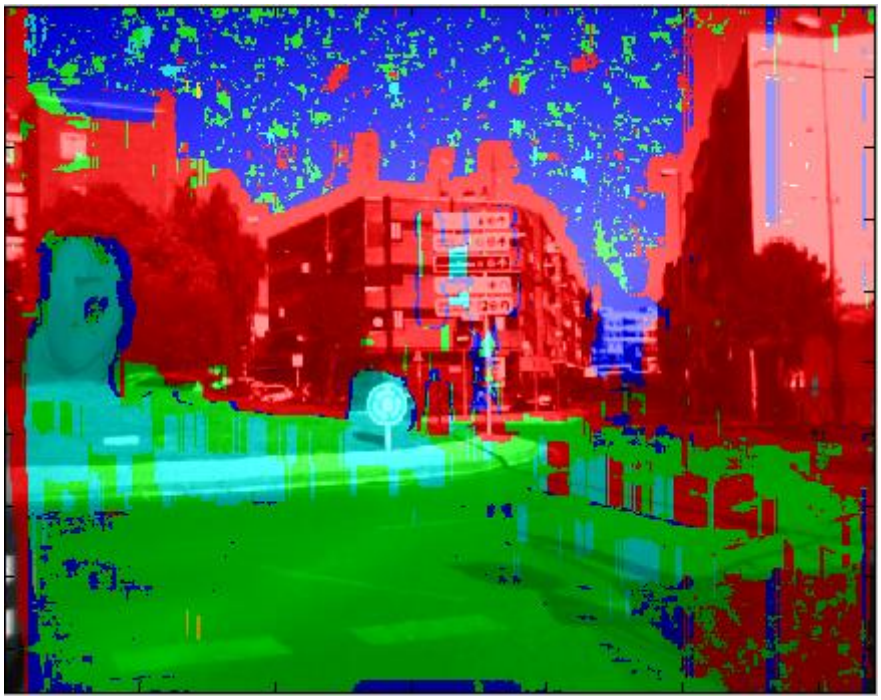


Figura 65 Resultados imagen (D) 30 neuronas

En esta imagen (Fig. 65) se podría decir que los resultados son prácticamente iguales a la anterior debido a que no tiene muchos objetos grandes. Cambian ciertos fallos de obstáculo por vehículo, pero no se podría afirmar una mejoría realmente destacable.

5.3. Tercera red neuronal.

Para esta red neuronal se usaran el mismo conjunto de datos, pero se utilizara el LBP extendido por el cual ahora tenemos 11 características por pixel. Como se ha aumentado el número de características notablemente la red neuronal constara de 85 neuronas ocultas. Al entrenar las red con información distinta, esto hace que los resultados tengan que cambiar más que solo aumentando el número de neuronas. Los resultados del entrenamiento para esta primera red con el LBP extendido son Fig. 66.

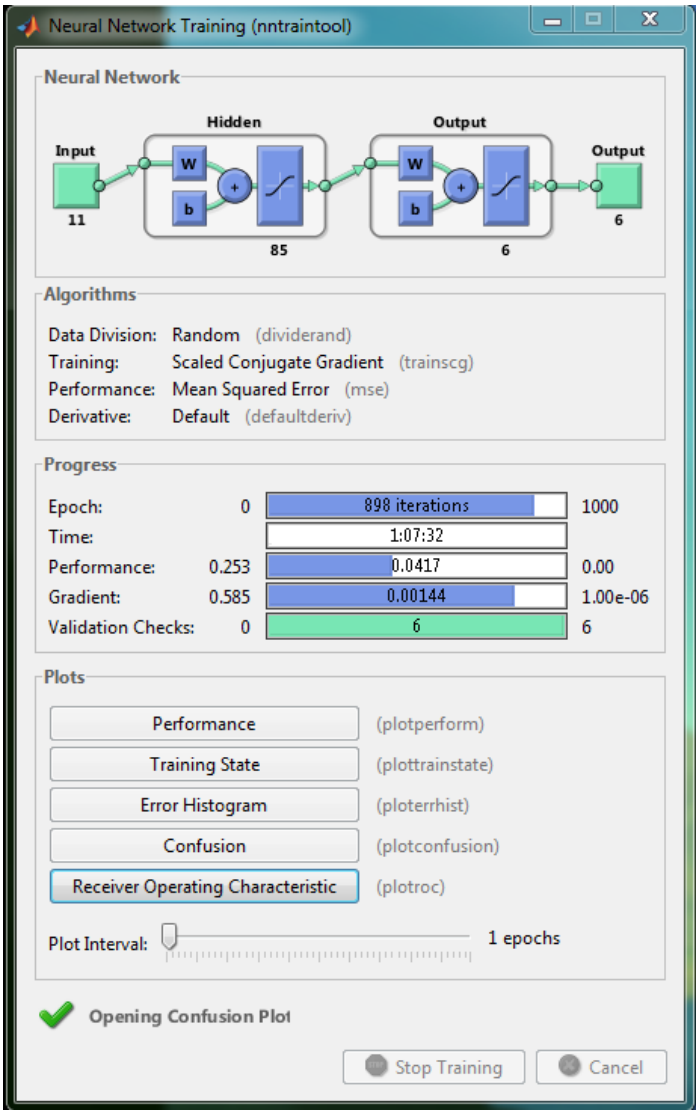


Figura 66 Entrenamiento red 85 neuronas

Para este entrenamiento Matlab ha realizado 898 iteraciones hasta completar las 6 validaciones de comprobación. Y el tiempo empleado han sido 1 hora, 7 minutos. Es un aumento considerable de tiempo, esto se debe a la cantidad de características, se ha pasado de 4 a 11 características por pixel, y al número de neuronas utilizadas.

La matriz de confusión de esta red neuronal es Fig. 67.

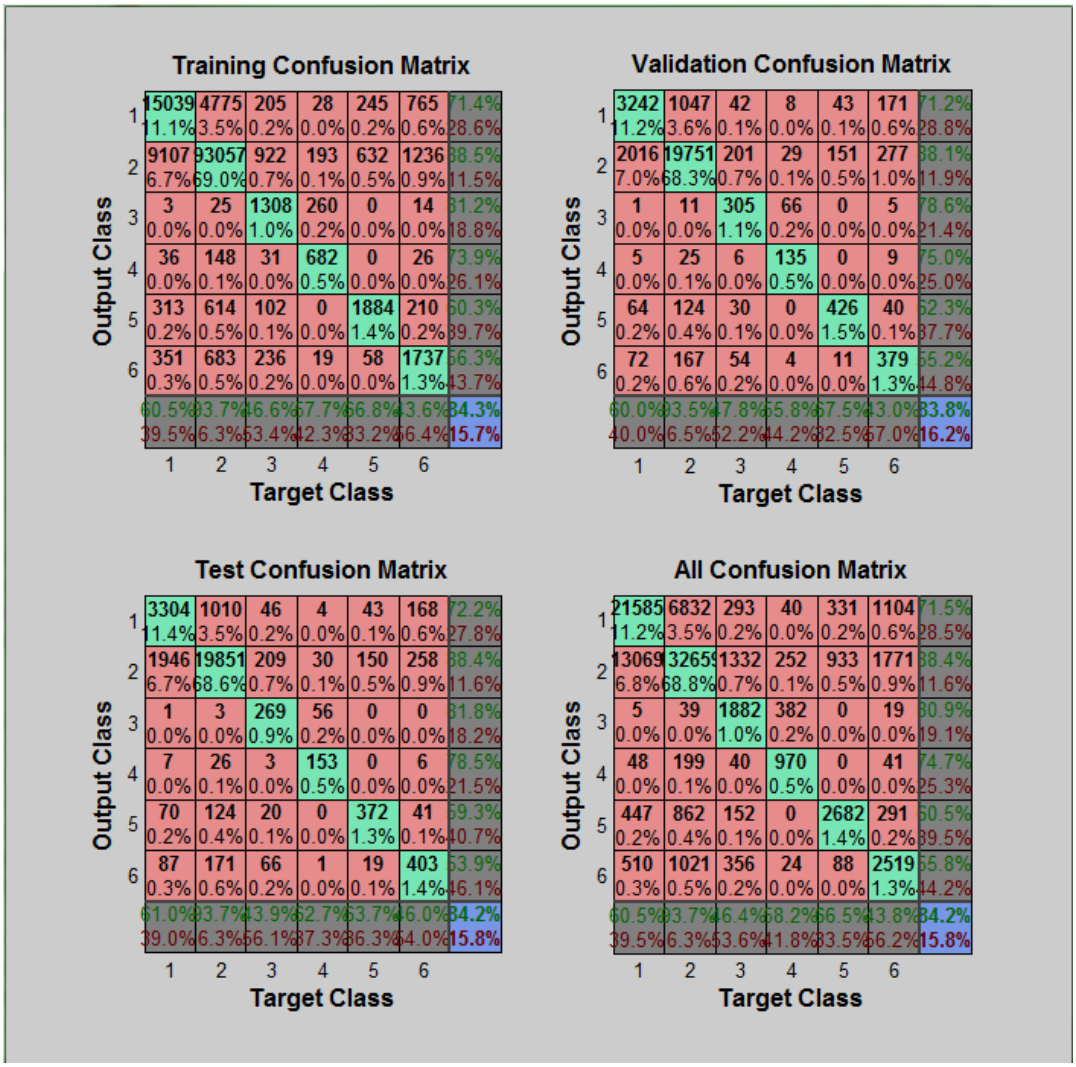


Figura 67 Matriz de confusión 85 neuronas

Los porcentajes arrojados por esta red no tienen por qué ser mayores que en los anteriores ejemplos, pero la identificación debería ser mejor al ofrecer más datos a la red. Los porcentajes obtenido son los siguientes:

- 1 = Oclusión: 72,2% de acierto
- 2 = Calzada: 88,4% de acierto
- 3 = Vehículo: 81,8% de acierto
- 4 = Peatón: 78,5% de acierto
- 5 = Cielo: 59,3% de acierto
- 6 = Obstáculo: 53,9% de acierto

Los porcentajes son mucho menores en todas las categorías pero veremos cómo influye los nuevos datos sobre los resultados.

Los resultados para la imagen (A) son Fig. 68.

Figura 68 Resultado imagen (A) 85 neuronas

Como se puede observar a simple vista los resultados de la red neuronal son peores que en los dos casos anteriores, el coche casi no se identifica, continua el error del muro como peatón, y hay ciertos objetos que antes eran obstáculos y que ahora son identificados como calzada. Esto se debe principalmente a la baja tasa de acierto obtenida en el entrenamiento.

Los resultados para la imagen (B) son Fig. 69.

Figura 69 Resultado imagen (B) 85 neuronas

Podemos observar que los resultados han empeorado en comparación, se puede observar como en la imagen anterior hay un sobre-etiquetamiento de calzada. Confundiendo gran parte de los obstáculos con calzada. Aunque la identificación de peatones sigue siendo bastante satisfactoria, esta red parece no tener el suficiente porcentaje de acierto.

Resultados imagen (C) son Fig. 70.

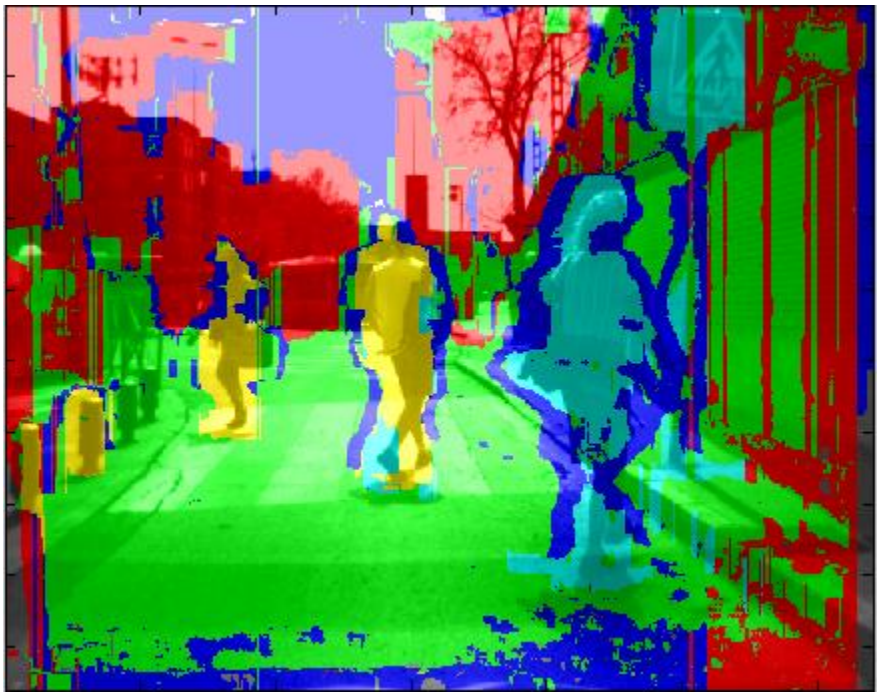


Figura 70 Resultado imagen (C) 85 neuronas

Se sigue repitiendo el mismo patrón de etiquetado erróneo.

Los resultados para la imagen (D) son Fig. 71.

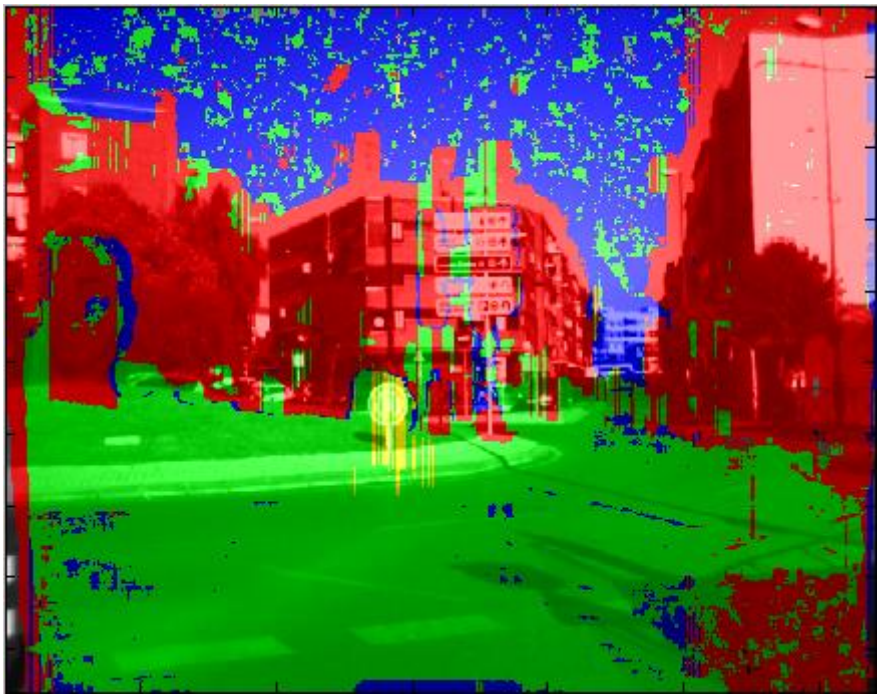


Figura 71 Resultado imagen (D) 85 neuronas

Se observa que en esta imagen se reduce mucho el error con respecto a los ejemplos anteriores, se han eliminado muchos errores de etiquetado como coche y también de peatones, también se han eliminado los errores que aparecían en la calzada. Solo hay un pequeño error con la señal que se etiqueta como persona, pero está totalmente justificado por los niveles de acierto.

Con esta red se intuye que es menos precisa y que tiende a eliminar los objetos pequeños, pero los resultados no son los suficientemente satisfactorios.

5.4. Cuarta red neuronal.

Para mejorar los porcentajes de la red neuronal anterior, se utilizara en este caso una red de 120 neuronas con los datos del LBP extendido. Los resultados de este entrenamiento son los siguientes Fig. 72.

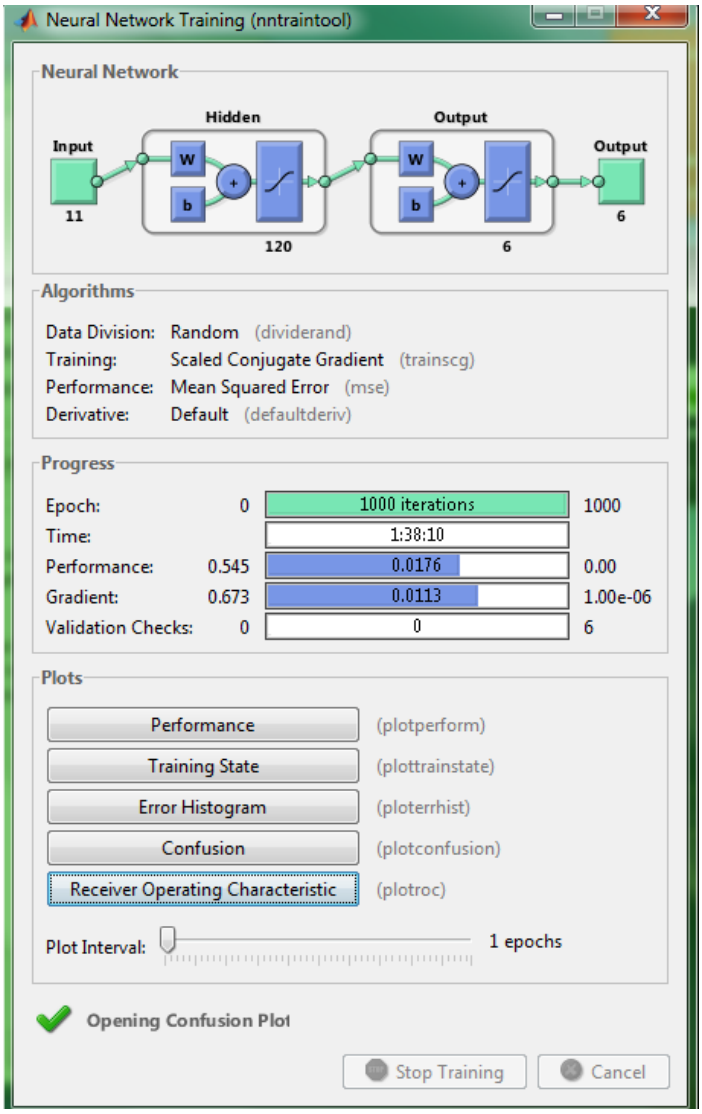


Figura 72 Entrenamiento red 120 neuronas

Como podemos observar en la Fig. 72 el entrenamiento ha realizado 1000 iteraciones, lo que quiere decir que el entrenamiento acaba porque ha llegado a su límite de iteraciones. No se han cumplido la 6 validaciones necesarias, pero esto no quiere decir que los resultados de la red no sean positivos, si no que Matlab considera que con 1000 iteraciones se tiene una red neuronal entrenada y lista para usarse. El tiempo empleado en este entrenamiento ha sido 1 hora, 38 minutos y 10 segundos. Y los resultados de la matriz de confusión son los siguientes Fig. 73.

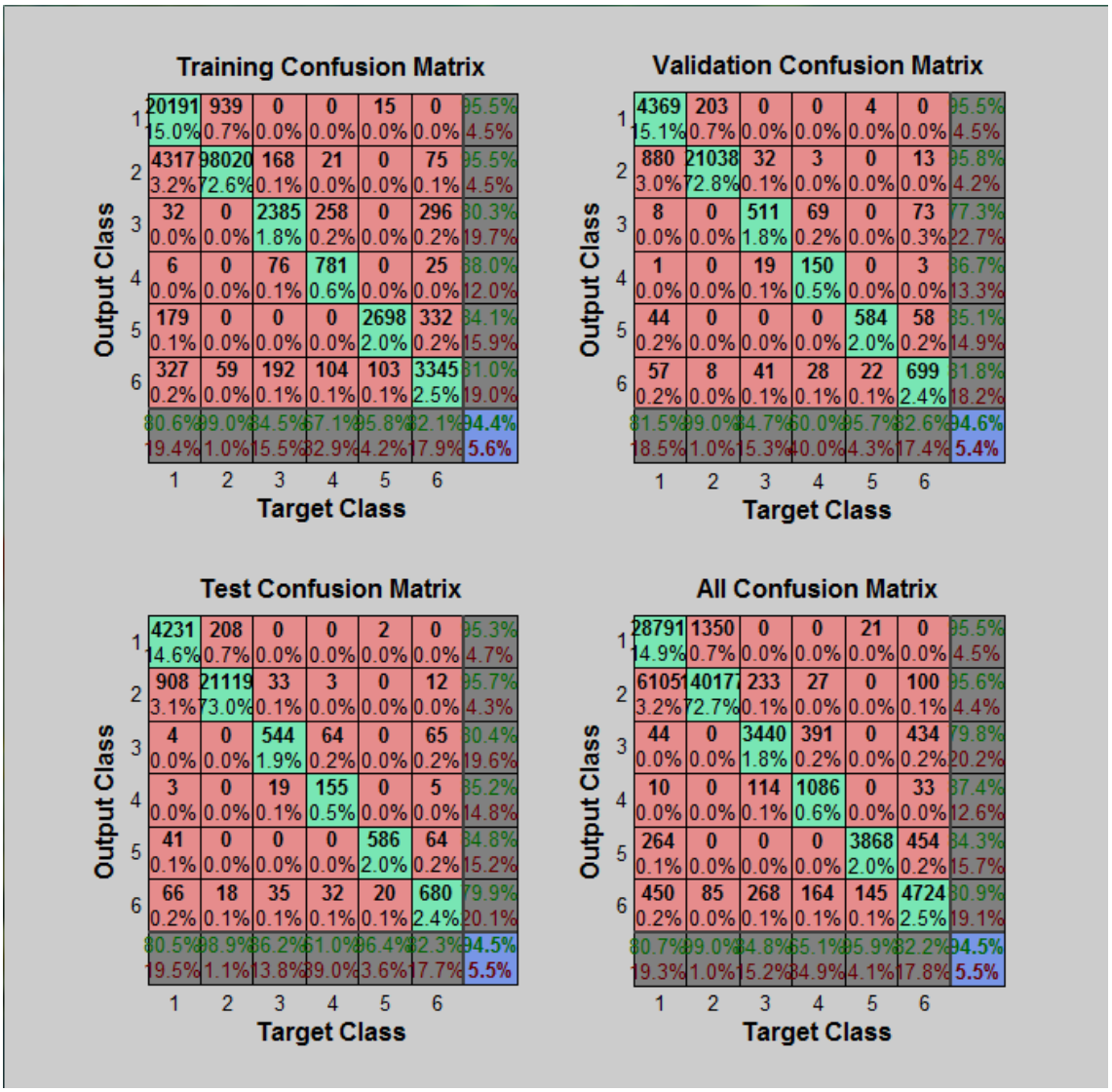


Figura 73 Matriz de confusión 120 neuronas

Los porcentajes de acierto para esta red son mucho más satisfactorios que en la red de 85 neuronas. Estos son:

- 1 = Oclusión: 95,3% de acierto
- 2 = Calzada: 95,7% de acierto
- 3 = Vehículo: 80,4% de acierto
- 4 = Peatón: 85,2% de acierto
- 5 = Cielo: 84,8% de acierto
- 6 = Obstáculo: 79,9% de acierto

Estos resultados son los más altos en porcentaje para todas las redes neuronales. A continuación se expondrá los resultados de esta red neuronal de 120 neuronas.

Resultados imagen (A), Fig.74.

Figura 74 Resultado imagen (A) 120 neuronas

Se aprecian muchas mejoras comparadas con el resultado de la red de 85 neuronas, pero se aprecia que el error del muro es persistente para todas las redes. El coche vuelve a estar bastante bien etiquetado pero sigue sin estar bien definido. Sigue habiendo una confusión entre obstáculos y coche o peatón. Pero los resultados de esta imagen se podrían calificar de aceptables sin tener en cuenta el error del muro.

Los resultados para la imagen (B), Fig. 75.

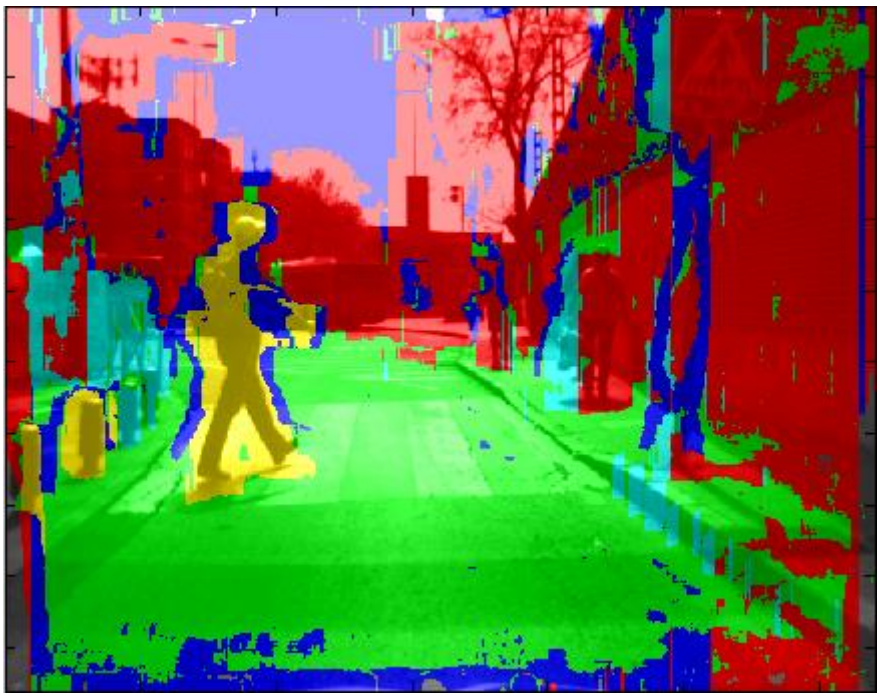


Figura 75 Resultados imagen (B) 120 neuronas

En esta imagen comparada con la red de 30 neuronas, observamos que se han corregido ligeramente el error del cubo de basura, que ahora tiene ciertas partes etiquetadas como obstáculo y menos como vehículo. Pero como se podía intuir de los resultados de la red de 85 neuronas, los detalles con este conjunto de características se pierden. Esto implica que se van a tener menos errores con los objetos pequeños, pero que esto también se van a perder y van a aparecer etiquetados como el objeto dominante alrededor de él, como se puede ver en el peatón que está situado a la derecha de la imagen. Este factor hace que haya que sopesar si se prefiere tener algo menos de error general o la detección de más objetos.

Los resultados para la imagen (C) son Fig. 76.

Figura 76 Resultados imagen (C) 120 neuronas

Los resultados son bastante similares a la red de 30 neuronas, pero se comete un fallo considerable, es la etiquetación por completo del peatón de la derecha como vehículo. Esto hace que se plante que esta red no es suficientemente fiable para el etiquetado de las imágenes.

Los resultados de la imagen (D) son Fig. 77.

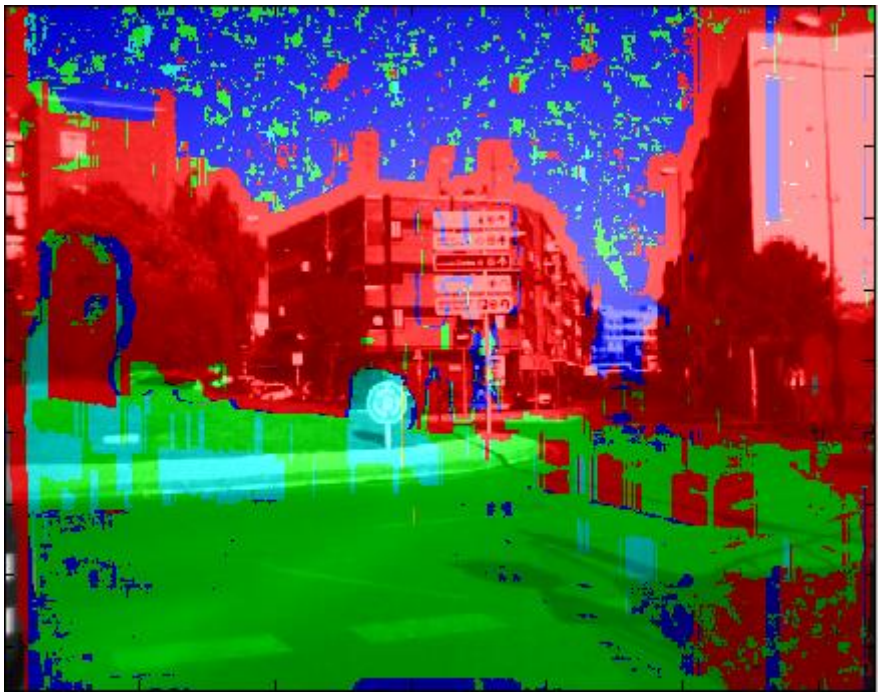


Figura 77 Resultado imagen (D) 120 neuronas

Los resultados de esta imagen son peores que para la red de 85 neuronas y muy similares a la red de 30, se sigue cometiendo cierto error con la etiquetación de parte de la rotonda como vehículo, pero se podrían incluir en el fallo por porcentaje de acierto. Pero ya no se comete el error de la estatua ya que la etiquetación es correcta.

Los resultados para esta red se pueden considerar satisfactorios ya que en la mayoría de los casos ha realizado un buen etiquetado de la imagen.

5.5. Tabla de resultados.

%	Oclusión	Calzada	Vehículo	Peatón	Cielo	Obstáculo
Red 10	98,2	95,6	69,1	73,7	81,5	71,9
Red 30	98,5	95,4	76	88,3	84,6	80,3
Red 85	72,2	88,4	81,8	78,5	59,3	53,9
Red 120	95,3	95,7	80,4	85,2	84,8	79,9

Tabla 2 Tabla de resultados

6. Conclusiones.

A la vista de los resultados anteriores se puede apreciar que las redes de 30 y 120 neuronas ofrecen unos resultados aceptables a la hora de la etiquetación. Las redes de 10 y 85 neuronas quedan descartadas debido a su bajo índice de acierto. Pero los resultados de las redes de 30 y 120 neuronas son lo suficientemente buenos como para una posible aplicación práctica.

De los resultados se podría decir que según que conjunto de datos utilicemos la red tendrá unas características u otras. Si se elige la red con LBP compacto (30 neuronas) se observa en sus resultados que se detectan objetos más pequeños, aunque estos estén mal etiquetados, la red es capaz de separarlos de los objetos que les están rodeando. Mientras que si se elige la red con el LBP extendido (120 neuronas) los datos son más precisos a grandes rasgos pero se pierden muchos detalles, y sigue habiendo un error de confusión en ciertas partes de la imagen. Todo esto lleva a la conclusión de que un LBP compacto ofrecerá más detalles, pero con ciertos errores de etiquetado, y con LBP extendido tendremos menos error general, pero se perderán objetos pequeños.

El error que se da en la imagen (A) en el muro es debido a que el entrenamiento de la red no hay suficientes ejemplos de entornos. Si la red neuronal constase del triple o cuádruple de puntos (Se necesitarían alrededor de 800.000 ó 1.000.000 de pixeles) y estos estuviese ponderados, es decir tener la misma cantidad aproximadamente de todas las etiquetas y el mayor número de entornos posible. Los resultados que arrojarían todas las redes serían muy superiores al actual.

También se podría mejorar las redes neuronales utilizando más descriptores o utilizando programas de depuración que fueran capaces de detectar objetos y si estos tienen más de una etiqueta, saber cómo se deben re-etiquetar los datos de ese objeto. De esta manera se conseguiría una mejora muy grande en los resultados finales.

Finalmente y a título personal yo elegiría como red la segunda red neuronal con 30 neuronas ocultas. Las razones para elegir esta red es que ofrece un buen compromiso entre detalles y etiquetado, es cierto que tiene cierto error de etiquetación, pero tiene la gran ventaja que es capaz de distinguir más objetos y de menor tamaño que la red de 120. Interesa que la red sea capaz de detectar objetos a gran distancia ya que nos estamos moviendo en un coche y nos interesa saber dónde esta los objetos que se mueven. Normalmente son de un tamaño pequeño como los peatones, que primero aparecen como objetos pequeños que cada vez se hacen más grandes e interesa ser capaz de detectarlos bien.

Esta aplicación para la conducción será muy útil en los próximos años, ya que la tendencia es la mejora de la seguridad al volante, y un sistema de aviso como este evitaría muchos accidentes con peatones y otros vehículos.

7. Costes

En este capítulo se realizara una estimación de los coste de este proyecto. Primero se expondrán las horas dedicadas a cada fase del proyecto:

Etapa	Tiempo
Comprensión y planteamiento del problema	10 horas
Documentación	30 horas
Redacción de la memoria (Teoría)	30 horas
Implementación del código	50 horas
Etiquetado de imágenes	60 horas
Entrenamiento de redes	30 horas
Pruebas y resultados	10 horas
Redacción de la memoria (Práctica)	30 horas
Total	**250 horas**

Con un coste por hora aproximado de 35€ para un programador junior, el precio total de obra de mano para el proyecto es:

$$250 horas * 35\ {}^{€}/_{horas} = 8.750€$$

EL coste de los materiales usados es:

Concepto	Coste
PC estándar	1000 €
Interfaz salpicadero	425 €
Cámara estéreo	3000 €
Licencia Matlab	6000 €
Otros materiales (Cables, etc)	100 €
Total	**10.525 €**

La instalación del material son 10 horas a 30 euros con lo que el coste son 300 €.

El coste total del proyecto es:

$$8.750€ + 10.525€ + 300€ = \mathbf{19.575}\ €$$

8. Bibliografía

[1] JIMENEZ MONJE, Violeta. Detección y localización de obstáculos en entornos urbanos mediante visión estéreo. Tutor: Basam Musleh Lancis. Universidad Carlos III de Madrid, Escuela politécnica superior.

[2] MINISTERIO DEL INTERIOR. Las distracciones son la causa de cuatro de cada diez accidentes.
<http://www.interior.gob.es/press/las-distracciones-son-la-causa-de-cuatro-de-cada-diez-accidentes-de-trafico-15244?locale=es>

[3] DIRECCIÓN GENERAL DE TRÁFICO. Información de accidentes.
<http://www.dgt.es/was6/portal/contenidos/documentos/seguridad_vial/estadistica/accidentes_24horas/resumen_anual_siniestralidad/resumen_siniestralidad043.pdf>

[4] WIKIPEDIA. Google driveless car.
<http://en.wikipedia.org/wiki/Google_driverless_car>

[5] CABELLO PARDOS, Emilio; CONDE VILDA, Cristina; RODRIGUEZ ARAGON, Licesio Jesús. Detección automática de conductas sospechosas en aeropuertos. Universidad Rey Juan Carlos.
<http://portal.uned.es/pls/portal/docs/PAGE/UNED_MAIN/LAUNIVERSIDAD/UBICACIONES/06/DUQUE_AHUMADA/PONENCIAS%20XX%20SEMINARIO%20DUQUE%20DE%20AHUMADA/2.PDF>

[6] BAMUELA, Luis. Visión por computador. Universidad politécnica de Madrid.
<http://www.dia.fi.upm.es/~lbaumela/doctorado/FormacionImagen.pdf>

[7] WIKIPEDIA. Cámara estenopeica.
<http://es.wikipedia.org/wiki/C%C3%A1mara_estenopeica>

[8] LECUMBERRY, Federico. Calculo de disparidad en imágenes estéreo, una comparación. Universidad de la Republica, Facultad de Ingeniería, Uruguay.

[9] WIKIPEDIA. Epipolar geometry.
<http://en.wikipedia.org/wiki/Epipolar_geometry>

[10] SCHOLARPEDIA. Local binary patterns.
<http://www.scholarpedia.org/article/Local_Binary_Patterns>

[11] SCHARSTEIN, Daniel; SZELISKI, Richard. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Middlebury College; Microsoft Corporation.
<http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf>

[12] WIKIPEDIA. Red neuronal artificial. <http://es.wikipedia.org/wiki/Red_neuronal_artificial>

[13] SENGUPTA, Sunando; GREVESON, Eric; SHAHROKNI, Ali; TORR, Philip H. S. Urban 3D semantic modelling using stereo visión. Oxford Brookes University.

[14] YANG, Yang; YANG, Jingyu; GUO, Dongyan. Pedestrian detection on moving vehicle. Nanjing University of science and technology; School of computer science and technology; School of computer and information engineering, Henan University, China.

[15] B. DOUILLARD; D. FOX; F. RAMOS; H. DURRANT-WHITE. Classification and semantic mapping of urban enviroments. The international journal of robotics research.

[16] BISHOP, Christopher. Pattern recognition and machine learning. Springer Verlag.

[17] B. PAYNE; A. TOGA. Surface mapping brain functions on 3d models. In Computer Graphics and Applications, 1990.

[18] L. LADICKY; C. RUSSELLl; P. KOHLI; P. H. TORR. Associative hierarchical crfs for object class image segmentation. In ICCV, 2009.

[19] Gavrila, D., Munder, S.: Multi-cue pedestrian detection and tracking from a moving vehicle. International Journal on Computer Vision 73(1), 41–59 (2007)

[20] MATLAB HELP.